THE UNIVERSITY OF GLAMORGAN
PRIFYSGOL MORGANNWG

School of Technology

THE USE OF NEURAL NETWORKS TO HELP FACILITATE THE ACCURATE PREDICTION OF ELECTRICITY DEMAND ON CRETE

Alexandros I. Manousakis

School of Technology
University of Glamorgan

A thesis submitted in partial fulfillment of the requirements of the University of Glamorgan for the degree of Master of Philosophy

October 2000

ACKNOWLEDGMENTS

I especially wish to thank my supervisors, Dr. George Papadourakis and Dr. Andrew Ware, for their intellectual support, guidance and advice. Their comments, suggestions and their patience in dealing with my errors have been more than valuable.

I would also like to thank the Local Power Corporation and especially Mrs A. Gigantidou for providing the necessary data and for their overall co-operation.

Finally, I would like to express my thanks to Dimitris Kotzinos.

Certificate of Research

This is to certify that, except where specific reference is made, the work presented in this thesis is the result of the investigation undertaken by the candidate.

Candidate .....

Director of Studies .....

Declaration

This is to certify that neither this thesis or any part of it has been presented or is being currently submitted for any other degree other that the degree of Master of Philosophy of the University of Glamorgan.

Candidate .....

LIST OF FIGURES 3
LIST OF TABLES 5
ABSTRACT 7
THESIS OUTLINE 8

1. INTRODUCTION 9
1.1. POWER SYSTEMS 9
1.2. THE POWER SYSTEM OF CRETE 11
1.3. MOTIVATION FOR RESEARCH 13

2. LITERATURE REVIEW 14
2.1. INTRODUCTION 14
2.2. LOAD FORECASTING MODELS 14
2.3. PREVIOUS APPROACHES 15
2.3.1. Time series approach 15
2.3.2. Regression approach 17
2.3.3. Other approaches 18
2.3.4. The neural networks approach 19
2.4. CONCLUDING REMARKS 23

3. NEURAL NETWORKS 25
3.1. INTRODUCTION 25
3.2. THE BIOLOGICAL MODEL 27
3.3. THE ARTIFICIAL MODEL 30
3.3.1. Introduction 30
3.3.2. Processing units 31
3.3.3. Connection between units 31
3.3.4. Network Topologies 33
3.3.5. Training of artificial neural networks 37
3.4. NEURAL NETWORK MODELS 39
3.5. MULTILAYER PERCEPTRONS 39
3.6. ERROR BACK-PROPAGATION ALGORITHM 41

4. DATA PREPROCESSING 44
4.1. INTRODUCTION 44
4.2. DATA GATHERING 44
4.3. DATA ANALYSIS 45
4.3.1. Data types 47
4.3.1.1. Numeric data representations 47
4.3.1.2. S 49
4.4. REJECTION OF FAULTY MEASUREMENTS 50
4.5. INPUT NORMALISATION 53
4.6. INPUT ENCODING 54
4.7. OUTPUT REPRESENTATION 59

5. EXPERIMENTS AND RESULTS 61
5.1. INTRODUCTION 61
5.2. NETWORK PARAMETERS 61
5.3. NETWORK RESULT EVALUATION 63
5.4. TRAINING INFECTIONS 64
5.4.1. Underfitting-Overfitting 64
5.4.2. Overtraining 64
5.5. EXPERIMENTAL CONDITIONS 65
5.5.1. The historical database 65
5.5.2. The neural network simulator used 67
5.6. NETWORK TRAINING 67
5.7. HOURLY LOAD DEMAND PREDICTION 68

5.7.1. The network architecture 68
5.7.2. The effect of the learning rate and momentum parameter 69
5.7.2.1. Learning rate equals 0.3, momentum parameter equals 0.9 69
5.7.2.2. Learning rate equals 0.6, momentum parameter equals 0.8 72
5.7.2.3. Adaptive learning rate and momentum parameter 73
5.7.3. The effect of the number of the training patterns 74
5.7.3.1. A reduction of the training patterns 74
5.7.3.2. A further reduction of the training patterns 75
5.7.4. The effect of the output representation 75
5.7.4.1. A neural network architecture without the bit encoded output 75
5.7.4.2. The effect of the number of the hidden neurons to a 16-output neural network 76
5.7.4.3. The effect of the learning rate and the momentum parameters to the 16-output neural network. 76
5.8. RESULTS' CONCLUSIONS-LEADS TO FURTHER INVESTIGATION 78
5.9. SEASONAL-SEPARATED NEURAL NETWORKS 83
5.9.1. A neural network for an hour ahead load prediction during the summer 83
5.9.2. A neural network for an hour ahead load prediction during autumn, winter and spring 84
5.10. DAILY SEPARATED NEURAL NETWORKS 85
5.10.1. Implementation of 7 different neural networks, one for each day of the week 87
5.11. FURTHER IMPROVEMENT OF THE RESULTS 88
5.12. A 24-OUTPUT NEURAL NETWORK FOR A WHOLE DAY LOAD FORECAST 92
5.13. NOON AND EVENING LOAD PEAKS FORECASTING 95
5.13.1. Noon peak prediction 96
5.13.2. Evening peak prediction 97

6. CONCLUSIONS AND FUTURE WORK 99

6.1. SYNOPSIS OF THE RESEARCH 99
6.2. CONCLUDING REMARKS 100
6.3. SUGGESTIONS FOR FUTURE WORK 102

APPENDIX A 104

A.1. THE PREPROCESSING SOFTWARE 104

APPENDIX B 107

B.1. SNNS NEURAL NETWORK SIMULATOR 107
B.2. TRAINING AND TEST PETTERNS' FORMAT 109

APPENDIX C 110

B.1. PUBLISHED PAPER 110

REFERENCES 120

FIGURE 1.1 WORLD GROWTH RATE FOR ELECTRICITY [2] 10

FIGURE 1.2 EVOLUTION OF THE LOAD DEMAND IN THE ISLAND OF CRETE [3], [4]. 12

FIGURE 3.1 (A) SCHEMATIC STRUCTURE OF A BIOLOGICAL NEURON, (B) SIMPLIFIED SKETCH OF SYNAPSES [38] 28

FIGURE 3.2 VARIOUS ACTIVATION FUNCTIONS FOR A UNIT 32

FIGURE 3.3 THE BASIC COMPONENTS OF AN ARTIFICIAL NEURAL NETWORK. THE PROPAGATION RULE USED 33

FIGURE 3.4 A FEEDFORWARD NETWORK WITH ONE INTERNAL LAYER OF NEURONS (A) 36

FIGURE 3.5 SUPERVISED LEARNING PARADIGM 37

FIGURE 3.6 UNSUPERVISED LEARNING PARADIGM 38

FIGURE 3.7 REINFORCEMENT LEARNING PARADIGM 39

FIGURE 4.1 ILLUSTRATION OF THE USE OF DATA PRE-PROCESSING AND POST-PROCESSING IN CONJUNCTION WITH A NEURAL NETWORK MAPPING. 46

FIGURE 4.2 AVERAGE DAILY LOAD DEMAND FOR THE YEARS 1994 AND 1995 51

FIGURE 4.3 AVERAGE DAILY LOAD DEMAND FOR THE YEARS 1994 AND 1995 AFTER THE REJECTION OF THE FAULTY AND ABNORMAL MEASUREMENTS 53

FIGURE 4.4 AVERAGE DAILY LOAD DEMAND FOR THE YEARS 1994 AND 1995 AFTER THE NORMALISATION TO THE BASE YEAR AND THE RESCALING TO [0,1] 54

FIGURE 4.5 WEEKDAY DEMAND NORMALISED IN [0,1] SPACE 56

FIGURE 4.6 HOURLY DEMAND NORMALISED IN [0,1] SPACE 57

FIGURE 4.7 MONTHLY DEMAND NORMALISED IN [0,1] SPACE 57

FIGURE 4.8 DEMAND PER MINIMUM TEMPERATURE VALUE NORMALISED IN [0,1] SPACE 58

FIGURE 4.9 DEMAND PER MAXIMUM TEMPERATURE VALUE NORMALISED IN [0,1] SPACE 58

FIGURE 5.1 ARCHITECTURAL GRAPH OF THE BASIC NEURAL NETWORK FOR THE HOURLY LOAD PREDICTION 69

FIGURE 5.2 MEAN SQUARE ERROR AS A FUNCTION OF THE LEARNING EPOCHS 70

FIGURE 5.3 PERCENTAGE OF THE ERROR OF THE NETWORK AS A FUNCTION OF THE LEARNING EPOCHS . 70

FIGURE 5.4 THE ACTUAL LOAD CURVE AND THE FORECASTED LOAD RANGE OF A TYPICAL DAY 72

FIGURE 5.5 AVERAGE DIFFERENCE OF THE PREDICTED LOAD VALUE FROM THE ACTUAL AS A FUNCTION OF THE LOAD RANGE FOR BOTH 4 AND 16 OUTPUT NEURONS. 78

FIGURE 5.6 ACTUAL AND PREDICTED LOAD VALUES SORTED IN ASCENDING ACTUAL LOAD VALUES 80

FIGURE 5.7 A DETAILED VIEW OF THE HIGH LOAD VALUES PRESENTED IN FIGURE 5.6 80

FIGURE 5.8 % DISTRIBUTION OF THE FAULTY PREDICTIONS PER MONTH 81

FIGURE 5.9 PERCENTAGE OF THE FAULTY LOAD DEMAND VALUES WHICH EXHIBIT THE LARGER DIFFERENCES FROM THE ACTUAL, BY A FACTOR OF 20MW OR GREATER AS A FUNCTION OF THE MONTH 81

FIGURE 5.10 % DISTRIBUTION OF THE FAULTY PREDICTIONS PER DAY OF THE WEEK 82

FIGURE 5.11 PERCENTAGE OF THE FAULTY LOAD DEMAND VALUES WHICH EXHIBIT THE LARGER DIFFERENCES FROM THE ACTUAL, BY A FACTOR OF 20MW OR GREATER AS A FUNCTION OF THE DAY OF THE WEEK 82

FIGURE 5.12 SYSTEM ARCHITECTURE FOR THE IMPLEMENTATION OF 7 DIFFERENT NEURAL NETWORKS, ONE FOR EACH DAY OF THE WEEK 86

FIGURE 5.13 DISTRIBUTION OF PATTERNS ACCORDING TO LOAD RANGE (STANDARD TRAINING DATABASE) 89

FIGURE 5.14 DISTRIBUTION OF PATTERNS ACCORDING TO LOAD RANGE (IMPROVED TRAINING DATABASE) 90

FIGURE 5.15 TEST PATTERNS' % OF FAULTY PREDICTIONS ACCORDING TO LOAD RANGE 91

FIGURE 5.16 ACTUAL AND PREDICTED LOAD VALUES SORTED IN ASCENDING ACTUAL LOAD VALUES. 91

FIGURE 5.17 ARCHITECTURAL GRAPH OF THE NEURAL NETWORK USED FOR 24-HOUR AHEAD HOURLY LOAD PREDICTION 93

FIGURE 5.18 ARCHITECTURAL GRAPH OF THE NEURAL NETWORKS USED FOR NOON AND EVENING PEAK LOAD FORECASTING 96

TABLE 4.1 RAW DATA FOR THE WEEK 01/06/1996 - 07/06/1996. EACH RECORD CONSISTS OF 24 LOAD VALUES OF THE HOURLY LOAD DEMAND, 2 LOAD VALUES FOR THE DAILY AND THE NIGHT PEAK, THE MINIMUM AN THE MAXIMUM TEMPERATURE. 45

TABLE 4.2 RAW RECORD DESCRIPTION 45

TABLE 4.3 GRAY CODE 49

TABLE 4.4 VARIOUS REPRESENTATIONS OF THE LOAD RANGES 60

TABLE 5.1 VARIABLE PARAMETERS OF MULTILAYER PERCEPTRON NEURAL NETWORKS 61

TABLE 5.2 AVAILABLE DATABASE SETS 66

TABLE 5.3 EXPLANATION OF TABLE 5.2 SYMBOLS 66

TABLE 5.4 SUMMARY OF THE INPUT VARIABLES USED TO THE NETWORK FOR THE HOURLY LOAD PREDICTION 68

TABLE 5.5 LOAD RANGES USED FOR THE PREDICTION 71

TABLE 5.6 NEURAL NETWORK RESULTS FOR LEARNING RATE EQUALS TO 0.3 AND MOMENTUM PARAMETER EQUALS 0.9 71

TABLE 5.7 NEURAL NETWORK RESULTS FOR LEARNING RATE EQUALS TO 0.6 AND MOMENTUM PARAMETER EQUALS 0.8 72

TABLE 5.8 NEURAL NETWORK RESULTS FOR ADAPTIVE LEARNING RATE MOMENTUM PARAMETER 73

TABLE 5.9 COMPARATIVE TABLE WITH THE RESULTS USING VARIOUS LEARNING RATE AND MOMENTUM PARAMETER. 74

TABLE 5.10 NEURAL NETWORK RESULTS WITH THE USE OF THE DIMINISHED TRAINING DATABASE (HTDB2) 74

TABLE 5.11 NEURAL NETWORK RESULTS WITH THE USE OF THE FEWEST TRAINING DATABASE (HTDB3) 75

TABLE 5.12 NEURAL NETWORK RESULTS WITH THE USE OF 16 OUTPUT NODES (2 HIDDEN LAYERS, 20 NODES EACH) 76

TABLE 5.13 NEURAL NETWORK RESULTS WITH THE USE OF 16 OUTPUT NODES (2 HIDDEN LAYERS, 30 NODES EACH) 76

TABLE 5.14 NEURAL NETWORK RESULTS WITH THE USE OF 16 OUTPUT NODES AND AN INCREASED LEARNING RATE PARAMETER. 77

TABLE 5.15 COMPARISON OF THE BEST PERFORMANCE ACHIEVED BY 4 AND 16 OUTPUT NEURONS RESPECTIVELY. 77

TABLE 5.16 NEURAL NETWORK RESULTS FOR LOAD PREDICTIONS DURING SUMMER FROM THE ACTUAL VALUES. 83

TABLE 5.17 NEURAL NETWORK RESULTS FOR LOAD PREDICTIONS DURING AUTUMN, WINTER AND SPRING 84

TABLE 5.18 TRAINING AND TEST SETS USED FOR THE IMPLEMENTATION OF THE 7 DAY-OF-THE-WEEK SEGREGATED NEURAL NETWORKS 86

TABLE 5.19 NEURAL NETWORK RESULTS FOR THE 7 DAY-OF-THE-WEEK SEGREGATED NEURAL NETWORKS 87

TABLE 5.20 PERCENTAGE OF THE FAULTY PREDICTED LOAD VALUES AS A FUNCTION OF THEIR DIFFERENCE FROM THE ACTUAL FOR THE DAY-OF-THE-WEEK SEGREGATED NEURAL NETWORKS 87

TABLE 5.21 NEURAL NETWORK RESULTS FOR LOAD PREDICTIONS USING THE IMPROVED TRAINING DATABASE. 90

TABLE 5.22 HOURLY LOAD PREDICTION FOR ONE DAY AHEAD. SUMMARY OF INPUT VARIABLES 92

TABLE 5.23 NEURAL NETWORK RESULTS FOR THE HOURLY LOAD PREDICTION FOR ONE DAY AHEAD, LEARNING RATE EQUALS 0.3, MOMENTUM PARAMETER EQUALS 0.9. 94

TABLE 5.24 NEURAL NETWORK RESULTS FOR THE HOURLY LOAD PREDICTION OF A WHOLE DAY AHEAD, LEARNING RATE EQUALS 0.5, MOMENTUM PARAMETER EQUALS 0.8. 94

TABLE 5.25 NEURAL NETWORK RESULTS FOR THE HOURLY LOAD PREDICTION OF A WHOLE DAY AHEAD - 30 NEURONS ON EACH HIDDEN LAYER. 94

TABLE 5.26 NEURAL NETWORK RESULTS FOR THE HOURLY LOAD PREDICTION OF A WHOLE DAY AHEAD - TRAINING WITH THE IMPROVED DATABASE 95

TABLE 5.27 SUMMARY OF INPUT VARIABLES USED FOR NOON PEAK LOAD FORECASTING 97

TABLE 5.28 NEURAL NETWORK RESULTS FOR THE NOON PEAK LOAD PREDICTION. 97

TABLE 5.29 SUMMARY OF INPUT VARIABLES USED FOR EVENING PEAK LOAD FORECASTING 98

TABLE 5.30 NEURAL NETWORK RESULTS FOR THE EVENING PEAK LOAD PREDICTION. 98

TABLE 6.1 SUMMARY OF THE LOAD FORECAST ERRORS 100

This thesis documents the research and analysis carried out in order to implement a system for short-term load forecasting for the isolated power system of the island of Crete, Greece.

The system was based on the use of multilayer perceptron neural networks. Data used to train and test the networks were obtained from the Public Power Corporation and span four years. These data were preprocessed using a special data preprocessing approach that is also introduced and described in this thesis. The data corresponding to the first year (1994) were used for the network training whereas the data for the other three years were used for testing. Extensive studies on the importance of various factors such as temperature, season, day of the week, *etc.* to the load demand were performed and the conclusions drawn lead to a better understanding of the load demand curve.

Various network topologies were validated so that their effect on the results could be evaluated and the best one chosen. This was done by studying the effect of factors such as learning rate, momentum, number of the training patterns *etc.* Also a new neural network output representation was utilized based on the use of Gray code, which provides a better error tolerance. The results show that the forecasted load average error achieved is extremely satisfactory and furthermore the majority of the erroneous predictions lie in the next output range (higher or lower).

The system is to be used with real-world data so as to provide the Public Power Corporation of Crete with the ability to forecast the load demand through the year.

Chapter 1 Provides an introduction to electric power systems, the power system of Crete, and a synopsis of the load forecasting problem before concluding with an explanation of the motivation that lead to the research. In **Chapter 2**, the classical load forecasting models and the previous approaches to the load forecasting problem are described. A detailed investigation of neural networks and an extensive description of the multilayer perceptrons and the error back-propagation algorithm are presented in **Chapter 3**. **Chapter 4** details the data preprocessing and the implementation of the input patterns database while **Chapter 5** illustrates the neural networks used, the experiments and the results. **Chapter 6** summarises the research conclusions and describes extensions for future work.

1. INTRODUCTION

1.1. Power systems

In 1882 Edison inaugurated the first central generating station in the USA. This had a load of 400 lamps, each consuming 83W. At about the same time, the Holborn Viaduct Generating Station in London was the first in Britain to cater for consumers generally, as opposed to specialised load. This scheme comprised a 60kW generator driven by a horizontal steam engine; the voltage of generation was 100V direct current [1]. The first major alternating-current station in Great Britain was at Deptford, where power was generated by machines of 10000h.p. and transmitted to consumers in London. During the same period, similar developments were taking place in the USA and elsewhere. Owing mainly to the invention of the transformer the advocates of alternating current prevailed, and a steady development of local electricity generating stations commenced, each large town or load center operating its own station.

World growth of electricity usage continued at about 7 per cent per year, implying a doubling every 10 years. The variation in rate of growth of electricity usage throughout the world and in the USA is shown in Figure 1.1. [2].

The vast electric energy systems that span every modern nation represent the largest and most expensive of man-made systems. The objective of an electric energy system is simply stated: "It shall generate electric energy in sufficient quantities at the most suitable generating locality, transmit it in bulk quantities to the load centers, and then distribute it to the individual customers in proper form and quality and at the lowest possible ecological and economic price" [44].

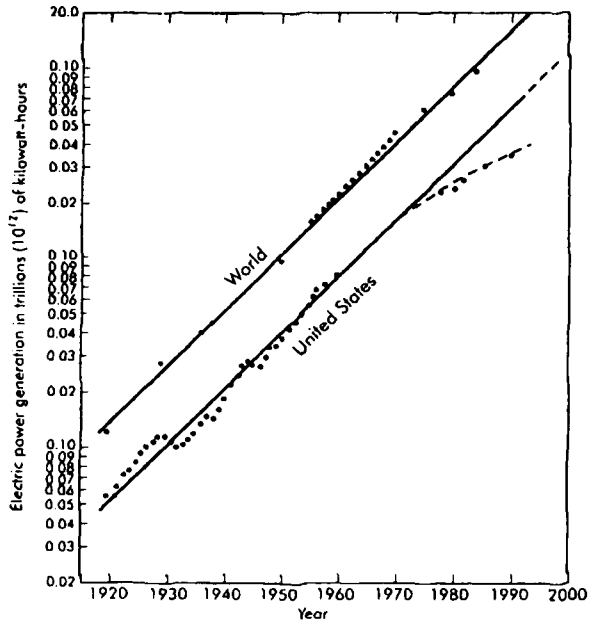


Figure 1.1 World growth rate for electricity [2]

A main concern in power distribution is to generate electricity to meet ever-changing demands for power. As electrical energy cannot be conveniently stored, its production and use are simultaneous. Hence the well-known supply-demand equation must always apply. It is no wonder, therefore, that the accurate prediction of power demand has become a necessity for efficient and profitable operation of electric utilities. The ability to accurately forecast electricity load demand offers significant potential for both cost savings through better planning and scheduling of generation resources, and for improvements in the safety and security of supply. In addition, the ability to forecast electric system load can also provide valuable information for possible energy interchange with other utilities. Accurate information for future loads is also useful in detecting many vulnerable situations in advance if applied to system security assessment problems. Forecast error in load prediction result in increased operating costs. On the one hand, underprediction of load result is a failure to provide the necessary reserves which translates to the increase of the probability of having a blackout or to higher costs due to the use of expensive peaking units. On the other hand, overprediction of load results in an unnecessary increase of reserves and hence operating cost.

The main objective of power system forecasting is to enable at all times an adaptation between demand and generation. This adaptation must consider load and generation characteristics and possible paths in transmission or distribution networks to supply energy to consumers. Load shapes must be represented by daily or weekly cycles. This representation is affected by some speculative and climatic variations.

Currently, there are three possible classifications for load forecasting: Short term forecasting: Prediction from the next hour to twenty-four hour ahead, medium-term forecasting: from the next day to the next week and long-term: forecasting beyond the next week.

1.2. The power system of Crete

Crete is an island far away from continental Greece, therefore Crete's power system is an isolated power system that uses energy production units independent from the power system of the rest of Greece. The island is lacking natural resources like water, coal, *etc.* so it is necessary to use oil as the moving power for the energy production plant. However, oil has numerous and important disadvantages compared to the other resources. The efficiency of the overall process is also poor compared to the oil cost. Whereas poor efficiency can be tolerated, the air pollution is a major problem that cannot be overlooked. Furthermore, the only production plant of the power system of Crete is located by the sea so the emitted pollution probably has destructive effects on the aquatic life.

The plant hosts 18 energy production units. The production capabilities range from 6 to 63 MW depending on the unit size. These units operate on a need-to-produce basis until the demand is satisfied or the total maximum production capability is reached.

At present, the daily load forecasts in the Power system of Crete are calculated manually. These manual forecasts are based on the assumption that the load curve for tomorrow is similar to the load curve of today. The weather influences are considered in the form of a rough addition but not calculated exactly. Therefore, the obtained results strongly depend on the experience of the load dispatcher.

The energy conservation that is accomplished by a programming system based on accurate load demand predictions is extremely valuable for the island. This value is enhanced by the substantial increase of the average annual load demand every year due to the increasing development factor at the island caused by industry and tourism, as illustrated in Figure 1.2 [3], [4]. This increase is larger during the period between May and September due to the population increase in the island related to the tourism industry, both workers and tourists. This results in some blackout incidents during the summer because the load demand exceeds the maximum possible power that can be produced by all units

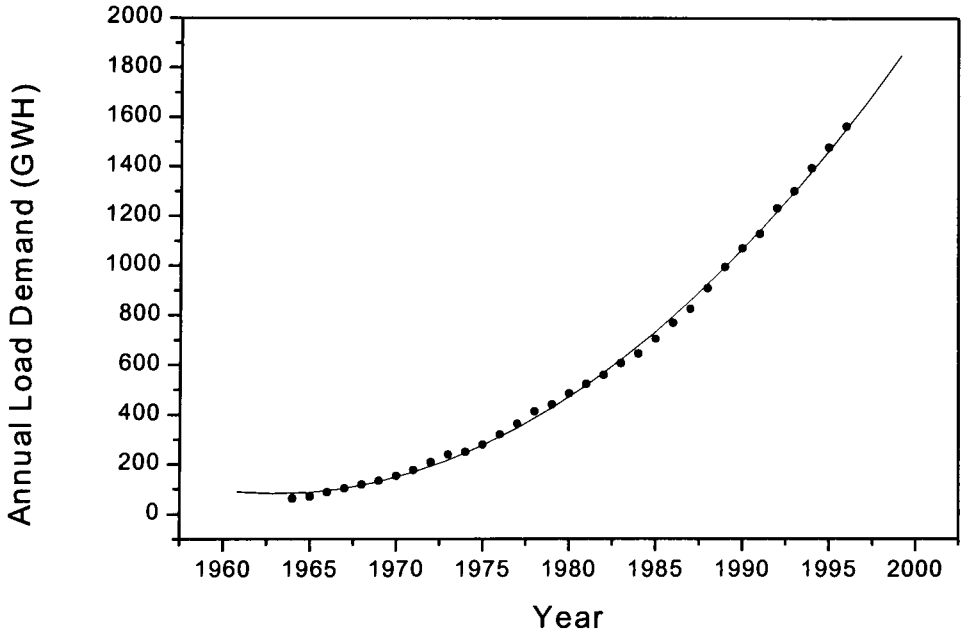


Figure 1.2 Evolution of the load demand in the island of Crete [3], [4].

1.3. Motivation for Research

In this research a new approach, based on artificial neural networks, to the load-forecasting problem of the isolated power system of island of Crete is presented. Extensive studies on the importance of various factors such as temperature, season, day of the week, *etc.* to the load demand are also performed.

The problem is largely divided into two basic parts:

- Data preprocessing, where a new data preprocessing methodology is introduced, and
- Construction of the neural network, where a new neural network output representation is utilized.

A variety of combinations of input patterns are used to examine the effect, in terms of the forecasting error, on the prediction of hourly and daily peak load.

Most of the previous approaches center on the forecasting of the average load. Although the latter is useful it can not provide a complete view of the load demand. In order to obtain meaningful results the peak load must be forecasted. In this research the forecasting of the average as well as the peak load is examined in order to improve the prognosis.

2. LITERATURE REVIEW

2.1. Introduction

Load forecasting is a well-documented problem. Many scientists around the world along with the power systems' managers have dealt with the problem for many years, introducing various techniques and methodologies. The results presented in many of the reports and papers dealing with the problem are not usually indicative of the quality of the work, due to the diversity of the nature of the different power systems. For example, the Power system in Taiwan exhibits a totally different behaviour to the Power System in Crete.

2.2. Load forecasting models

The classical load forecasting models consist, mathematically, of the following four components [23]: basic load; time dependent components (season, month, day of the week, hour); weather-dependent component (temperature, wind speed, sunshine, humidity *etc.*); noise or residual component and special effect correction. The basic load is the largest component accounting for 90-95% of the total load. The weather dependent component accounts for 5-10% while the noise or residual component accounts for no more than 2-3%. The effect of the time dependent components depends on the power station itself, the social habits *etc.* The special effect correction is normally zero and comes into play only under special and unusual conditions (such as unusual weather patterns, national holidays, abrupt weather change, shutdowns, strikes in large industry companies, political and economical effects).

2.3. Previous approaches

According to the above model there is a fully non-linear functional correlation between all the above factors and the load patterns. This relationship has to be represented to achieve the capability for an efficient prediction.

Previous approaches to electric load forecasting fall, in general, into three categories in accordance with the techniques they employ [33]. The first approach treats the load pattern as a time series problem and the future loads are predicted by using series analysis techniques [5-13]. The second approach recognises that the load pattern is highly correlated with weather variables, and finds a functional relationship between weather variables and the system load. The future load can then be forecast by inserting the predicted weather information into the predetermined functional relationship [14-21]. The third approach is the use of artificial neural networks [23-34] that combines both time series and regression approaches.

2.3.1. Time series approach

General problems with the time series approach include the inaccuracy of prediction and numerical instability. One of the reasons this method often gives inaccurate results is that it does not utilise meteorological information. The time series approach mostly utilises computationally cumbersome matrix-oriented adaptive algorithms, which, in certain cases, may be unstable. The time series approach is based on the presumption that the load pattern is nothing more than a time series signal with known seasonal, weekly, and daily periodicities. These periodicities give a rough prediction of the load at the given season, day of the week and time of the day. The difference between the prediction and the actual load can be considered as a stochastic process. Analyzing the random signal may lead to more efficient prediction. The techniques used for the random signal analysis include Kalman filters [11-13], the Box-Jenkins method [5-6], auto-regressive moving average (ARMA) model [8] and spectral expansion technique [7].

Kalman Filters are a part of the linear filter theory and they are based on a linear combination of a model function with unknown coefficients. It is a technique used for estimating or predicting the next state of a system based upon a moving average of measurements driven by additive white noise. The Kalman Filter estimates the next predictions of load by the usage of the last forecast errors (the errors between predicted and measured values). With the use of the last measurement, included in the forecast error, the Kalman Filter calculates the load prediction and therefore adapts itself more rapidly than conventional regression algorithms. However, with the use of the last measurements to calculate the new prediction, Kalman-Filters seem to be fit only for short-term load forecasting problems. Another critical problem that Kalman Filters have is the determination of the optimal weighting of the given inputs, because they influence directly the forecasting error. The determination of the optimal weights is a major task for the development of Kalman Filters and very time consuming.

The Box-Jenkins method requires the autocorrelation function for identifying proper ARMA models. This can be accomplished by using pattern recognition techniques. A major disadvantage of this technique is its slow performance [8].

The ARMA model is used to describe the stochastic behaviour of hourly load patterns on a power system. The ARMA model assumes the load at the forecasting hour can be estimated by a linear combination of the previous few hours. Generally, the larger the data set, the better is the result in terms of accuracy. However, a longer computational time for the parameter identification, is required.

The spectral expansion technique utilises the Fourier Series. Since load patterns can be approximately considered as a periodic signal, a load pattern can be decomposed into a number of sinusoids with different frequencies. Each sinusoid with a specific frequency represents an orthogonal base [9]. A linear combination of these orthogonal bases with proper coefficients can represent a perfectly periodic load pattern if the orthogonal basis span the whole space. However, load patterns are not perfectly periodic. This technique usually employs only a small

fraction of possible orthogonal bases set, and therefore is limited to slowly varying signals. Abrupt changes of weather cause fast variations of load pattern, which result in high frequency components in the frequency domain. Therefore, the spectral expansion technique can not provide any accurate forecasting for the case of fast weather change unless a sufficiently large number of base elements are used.

Generally, techniques in time series approaches give efficient results unless there is an abrupt change in the environmental or sociological variables, which are believed to affect load pattern. If there is any change in these variables, the time series technique is no longer useful. On the other hand, these techniques use a larger number of complex relationships, require a long computational time and result in possible numerical instabilities.

2.3.2. Regression approach

Most regression approaches try to find functional relationships between weather variables and the load demands. The general procedure for the regression approach is: select the weather variables; assume basic functional elements; and find proper coefficients for the linear combination of the assumed basic functional elements.

Since temperature is the most important information of all weather variables, it is used most commonly in the regression approach. Most regression approaches have simple linear or piece-wise linear functions as the basic functional elements [14-19]. An example of modelling the temperature-dependent component is given by the following formula:

$$L = \sum_{i=1}^{N} a_i T\{U(T - T_{i1}) - U(T - T_{i2})\} + C \qquad (2.1)$$

where $U(T) = \begin{cases} 1, if T \geq 0 \\ 0, if T < 0 \end{cases}$, a_i, T_{i1}, T_{i2}, and C are constants,

and $T_{i1}>T_{i2}$ for all *i*.

The variables (*L*, a_i, *T*, T_{i1}, T_{i2}, *and C*) are temporally varying. The time-dependency, however, is not explicitly noted for reasons of notational compactness.

After the basic functional forms of each subclass of temperature range are decided, the proper coefficients of the functional forms are found in order to make a representative linear combination of the basic functions. The functional relationship between load and weather variables, however is not stationary, but depends on spatio-temporal elements. The conventional regression approach does not have the versatility to address this temporal variation. It, rather, will produce an averaged unit. Therefore, an adaptable technique is needed.

2.3.3. Other approaches

Approaches other than regression have been proposed for finding functional coefficients.

Lu *et al* [20] utilise the modified Gram-Schmidt orthgonalization process (MGSOP) to find an orthogonal basis set which spans the output signal space formed by load information. The MGSOP requires a predetermined cardinality of the orthogonal basis set and the threshold value of error used in adaptation procedure. If the cardinality of the basis set is too small or the threshold is not small enough, the accuracy of the approach suffers severely. On the other hand, if the threshold is too small, numerical instability can result. The MGSOP also has an ambiguity problem in the sequence of input vectors. Different exposition of input vectors result in different sets of orthogonal basis and different forecasting outputs.

Jabbour *et al.* [10] used a pattern recognition technique to find the nearest neighbour for the best eight hourly matches for a given weather pattern. The corresponding linear regression coefficients were used.

An application of the Generalized Linear Square Algorithm (GLSA) was proposed by Irisarri *et al.* [17]. The GLSA, however, is often faced with numerical instabilities when applied to a large data set.

Rahman *et al.* [16], Torres *et al.* [35], Park *et al.* [36] have applied an expert system approach. The expert system takes advantage of the expert knowledge of the operator. It makes many subdivisions of the temperature range and forms different functional relationships according to the hour of interest. It shows fairly accurate forecasting. Expert system models are discrete and logical in nature, and use the knowledge of a human expert to develop rules for forecasting. However, transforming the knowledge of an expert to a set of mathematical rules is often very difficult.

While each of these approaches demonstrates considerable success in forecasting accuracy, they are all subject to the risk that future experience will not resemble past experience. This, of course, is the risk associated with the fundamental assumption of all statistical modelling approaches. Some modelling approaches, such as Box-Jenkins have been used to update a change in the most recent experience. Nonetheless, the bias introduced by the large historical database is not eliminated by adaptive updating procedures.

2.3.4. The neural networks approach

The multilayer artificial neural networks possess a number of properties, which make them an attractive alternative choice. The advantage of artificial neural networks over the statistical models lie in their ability of realisation of the complex non-linear relationship between the predicted values and indicators such as historic data (data collected in the past) and exogenous factors. The complex relationship between indicators (historical data and exogenous factors) and predicted values can be too hard to capture in expert system rules or analyse reliably with traditional statistical methods. Artificial neural networks can be trained to form a nonlinear model and use this model to generalise on new cases that are

not part of the training data. Artificial neural networks can provide very promising results for this type of problem because of the following facts:

- availability of a huge amount of historical data,

- complex relationship between the indicators and the load values to be predicted,

- possibility to include exogenous variables into one model.

The evaluation of neural network approach is interesting from another point of view. Following the terminology in [23], the design features of short-term load predictors include:

a) Adaptiveness. Parameters in the forecast model are not stationary and must be changed over time.

b) Recursiveness. As new data become available forecast model should react appropriately to new data, re-estimation of the parameters is required.

c) Computational economy. Algorithms need to be economical with respect to the execution time.

d) Robustness. The model should produce a reasonable forecast when a part of the time series is locally mis-specified, measurement error appearing in the data or default value must be used through system or operator failure.

Artificial neural networks, under certain limitations can sufficiently respond to all the above criteria therefore they are very promising for electric load forecasting. They offer the potential to overcome the reliance on a functional form of the forecasting model

Recently, a lot of interest has been focused on the applications of artificial neural networks in load forecasting [23-34]. In these studies, neural network techniques have been applied to correlate all the parameters that affect the load demand, to forecast the daily peak load, total load and hourly load. Several types of neural network architectures are used. The feedforward multilayer perceptron is the most popular structure in time series forecasting. Extensive studies have been carried out on the selection of the inputs to the neural network.

B.Bitzer *et al.* [31] compared several methods such as Kalman-Filters, regression algorithms and neural networks, to predict the load demand in the isolated power supply system of Crete. In total 28 different neural networks were used, four for every day of the week, using four different data sets in order to consider the seasonal load trends. They also examined two different input vector topologies. The first input vector is composed of 56 inputs as follows: the hour of the predicted load, predicted temperature of the current day (minimum/maximum), average temperature of the last three days, the last 48 measurements of the loads, the load of the last day at the same hour, the load of the last week at the same hour. They then reduce the input vector with the aim to increase the performance of the networks. Therefore the neural network have been trained with the following 17 inputs: predicted temperature of the current day (minimum/maximum), average temperature of the last three days, the last 6 measurements of the loads, the load of the last day at the same hour and the previous two loads, the load of the last week at the same hour and the previous two loads. Neural networks with one hidden layer with 10 neurons and one neuron in the output layer were used. They used different learning rules all based on the back-propagation learning rules. They reached the best performance using the seasonal trained neural network with the 55 input vectors. They also found out that the Kalman-Filter adapts rapidly to the seasonal trends but it was not able to predict the loads over a longer period.

Kiartzis *et al.* [24] used a neural network capable of forecasting the next 24 hour load profile. The network architecture that finally was selected had 63 input neurons, 63 hidden neurons and 24 output nodes. The input vector is composed

as follows: 24 values which represent the load one day before the forecasting day, 24 values which represent the load two days before, minimum and maximum temperature in two weather stations one day before, minimum and maximum temperature forecast in both weather stations, 7 nodes for the days of the week. The error back-propagation learning rule was used to train the network and the forecast error achieved was close to 2.6%.

Yuan-Yih Hsu *et al.* [25] used two different neural networks to achieve load forecasting. An artificial neural network based on unsupervised self-organising future maps has been developed for identifying those days which have similar hourly load patterns and belong to the same day type. The hourly load pattern of a day is obtained by averaging all those load patterns. In addition to the prediction of the hourly load pattern, daily peak and valley load must be predicted before the hourly loads can be forecasted. Thus, a three-layer feedforward neural network is developed with 46 nodes in the input and in each of the hidden layers and one output node. It is concluded from the study results that the neural networks can achieve accurate forecasting of hourly loads in a very efficient manner.

K.Y.Lee *et al.* [30] also used a two layer neural network for one day ahead load forecasting. The load patterns were classified into weekday patterns and weekend-day patterns. Several network topologies were used. The best results were achieved with the use of 48 input neurons, which are the latest 48 hours load data, 90 hidden and 24 output nodes.

Another method also was studied by separating the load forecasting into 3 parts which results in the reduction of the network size. The network, with the use of this method, was composed of 8 input nodes: 3 inputs were used for the load in the three day parts one day before the forecasted day, 3 inputs for the load two days before and 2 inputs which represent the forecasted load of the previous 2 parts of the forecasted day. The back-propagation learning algorithm was used for the training of the network. The second approach gave faster convergence and better results.

Shin-Tzo Chen *et al.* [26] used a nonfully connected neural network for short term load forecasting. The model consists of one main neural network and three supporting networks. The main network was used to provide the model's basic forecast reference and the supporting networks to increase the learning capacity of the proposed model. The 31-node input vector of the main network consists of the following: three nodes which represent the load from the previous three hours which occur prior to the forecasting day, three nodes from the same three hours of a day before, six inputs for the loads from the same hours of the previous week, three nodes for the average daily temperature of the forecasting day, a day before and two days before, and three for the last three hour temperatures, seven inputs for the day of the week, five inputs which represent the forecasting hour in binary representation. 41 hidden neurons are used in the hidden layer and 42 nodes in the output layer. They achieved 1.12% peak error.

A two-layer back-propagation model is used by Kermanshahi *et al.* [33] for forecasting daily loads of a Canadian electric utility. They used 24 neurons as the output, 86 neurons as the input and 20 hidden neurons were selected for the hidden layer. The obtained results are compared with the present linear regression technique for load forecasting in the Power system and showed an improvement.

Baumann *et al.* [34] presented a load forecasting method based on Kohonen's self-organising feature maps and a one layered linear delta-rule artificial neural network. Their method includes three parts: First, the self-organised arrangement of reference days in Kohonen maps, second the auto-associative memory part with trend correction and third the weather dependent load correction. The errors vary between 1.5% and 1.7% depending on the season.

2.4. Concluding remarks

Several types of neural network architectures were used in the electric load forecasting problem including feedforward multilayered perceptron and Kohonen

self-organising feature maps. The results shows that the artificial neural networks are suitable to interpolate among the load and temperature pattern data of the training sets to provide the future load pattern. Compared to the other methods, the artificial neural networks allow more flexible relationships between temperature and load patterns.

3. NEURAL NETWORKS

3.1. Introduction

The mysteries of the human mind have fascinated scientists and philosophers for centuries. The brain is very efficient in accomplishing such complex tasks as learning and optimisation. This is the reason artificial neural networks are, in a loose sense, based on concepts derived from research into the nature of the brain.

Everyday observation shows that the modest brains of lower animals can perform tasks that are far beyond the range of even the largest and fastest modern computers. Just imagine that any mosquito can fly around at great speed in unknown territory without bumping into objects blocking its path. A frog's tongue can catch insects in full-flight within a split second. No present-day computer has sufficient computational power to match these and similar accomplishments. They typically involve some need for the recognition of complex optical or acoustical patterns, which are not determined by simple logical rules.

Great efforts were made in the last decades to solve such problems on traditional programmable computers. One product of these efforts was the emergence of the techniques of artificial intelligence and the techniques of artificial neural networks. While the products of artificial intelligence, which are more appropriately described as expert systems, had a number of impressive successes, e.g. in medical diagnosis, they are much too slow to perform the analysis of optical or speech patterns at the required high rate. Moreover, the concept of artificial intelligence is based on formal logical reasoning, and can thus only be applied when the logical structure of a certain problem has been analysed. Another source of dissatisfaction is that the speed of solid-state switching elements, the

basic units of computers, has reached a point where no increase by orders of magnitude can be expected in the near future.

Neural networks seem to be particularly well suited to approaching problems in subsequent steps of increasing detail, *i.e.* in hierarchical steps. The emphasis here is more on the geometrical or topological aspects of the problem than on logical relations between its constituent parts. For this reason the term geometrical, as opposed to the usual analytical, approach to computing has been coined. An important aspect of problem solving in hierarchies is that the network is potentially capable of *classification* and *generalisation.* Although traditional methods of expert systems can readily solve these tasks if the classification criteria or generalisation rules are known, they have difficulty establishing such relations on their own. This is different in the case of neural networks, which have been shown to be able to learn and generalise from examples without knowledge of rules.

At the present state of development, traditional computers are still superior to neural networks in performing tasks or solving problems whose structure has been thoroughly analysed. However, it is only fair to state that often many man-years of analysis are required before the problem is sufficiently well understood to be accessible to computational treatment. On the other hand, a neural network may be able to learn from a large number of examples within a relatively short period of synaptic-strength adjustment. Another potentially important advantage of neural networks is their high degree of error resistivity. A normal computer may fail completely in its operation if only a single bit of stored information or a single program statement is incorrect. This leads to a high risk of failure, especially if the system cannot be continuously supervised. By contrast, the operation of a neural network often remains almost unaffected if a single neuron fails, or if a few synaptic connection break down. It usually requires a sizeable fraction of failing elements before the deterioration is noticeable. In view of the rapidly increasing use of electronic data processing in vital areas this is an attractive feature of neural computing, which may become of practical importance.

A major disadvantage of neural networks is the broad lack of understanding of how they actually solve a given cognitive task. Neural networks do not break the problem down into its logical elements but rather solve it by a holistic approach, which is hard to penetrate logically. When the logical structure of the solution of the problem is not known anyway, this represents an advantage, but it becomes a disadvantage when one wants to determine whether the solution provided by the neural network is correct. No one knows how to judge the performance of a neural network knowing only its architecture, and it is almost impossible to determine what task the network performs from the pure knowledge of the synaptic weights.

3.2. The Biological Model

It is known from neurophysiology that the human brain contains a massively interconnected network of between 10^{10} and 10^{11} neurons of nerve cells [37]. A very simplified schematic view of a biological neuron is shown in Figure 3.1a [38]. In this schematic model, the following fundamental parts can be recognised:

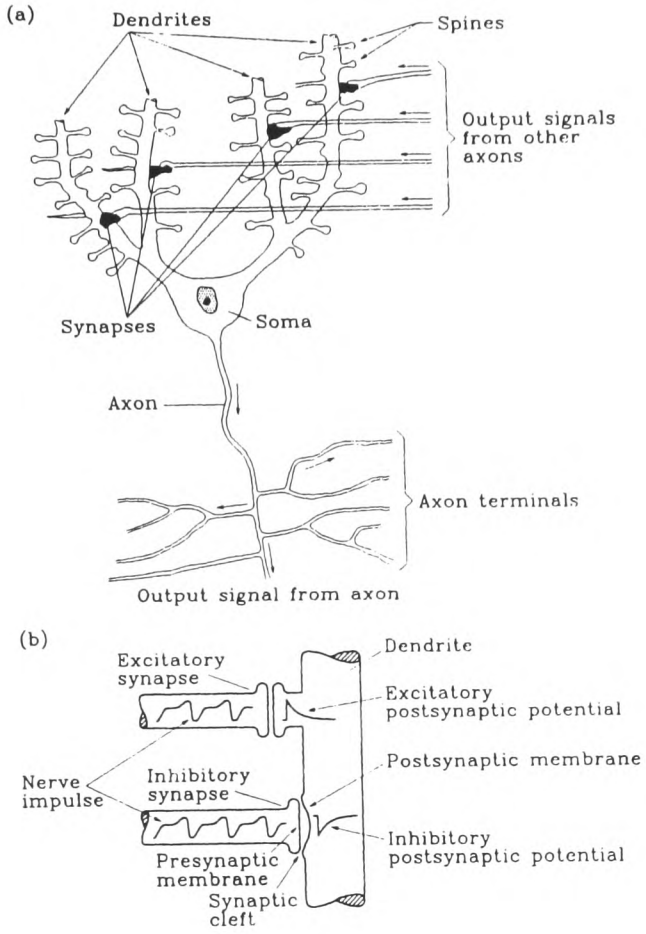


Figure 3.1 (a) Schematic structure of a biological neuron, (b) Simplified sketch of synapses [38]

1. The *soma or body cell* is the large, round central body in which almost all the logical functions of the neuron are realised. The body cell contains the genetic and metabolic machinery necessary to keep the neuron alive. The neuron soma contains the nucleus and the protein-synthesis machinery.

2. The *axon* (output) is the nerve fibre attached to the soma and can serve as the final output channel of the neuron. An axon is usually highly branched. The initial segment of the axon is called an axon hillock. There the signals are generally converted into sequences of nerve pulses, which are propagated without attenuation along the axon to the target cells.

3. The *dendrites* (inputs) represent a highly branching tree of fibres. These long irregularly shaped nerve fibres are attached to the soma. There are 10^3 to 10^4 dendrites per neuron. Dendrites connect the neuron to a set of other neurons. Dendrites either receive inputs from other neurons via specialised contacts called *synapses* or connect other dendrites to the synaptic outputs. Generally dendrites are regarded as providing receptive surfaces for input signals to the neurons and as conducting signals with decrement to the body cell and the axon hillock.

4. *Synapses* are specialised contacts on a neuron, which are the termination points for the axons from other neurons. Synapses play the role of interfaces connecting some axons of the neurons to the spines of the input dendrites. Synapses are capable of changing a dendrite's local potential in a positive or negative direction (Figure 3.1b). Due to their function the synapses can be of excitatory or inhibitory nature according to their ability to increase or damp the neuron excitation. Storing of information in a neuron is supposed to be concentrated and strengths (weights) of the synaptic connections. The human synapses are mainly of a very complex chemical nature while synapses in nervous systems of primitive animals, such as insects, are predominantly based on electrical signal transmission.

According to the above simplified model of the neuron, the body cell receives inputs from other neurons through adjustable or adaptive synaptic connections to dendrites. The output signal (consisting of nerve impulses) from a cell is transmitted along a branching axon to the synapses of other neurons. When a neuron is excited it produces nerve impulses which are transmitted along an axon to the synaptic connections of other neurons. The output pulse rate depends on both the strength of the input signals and the weight or strength of the corresponding synaptic connections. The input signals at the excitatory synapses increase the pulse rate while the input signals at the inhibitory synapses reduce the pulse rate or even block the output signal.

A neuron operates in mixed digital and analogue form. The information between neurons is transmitted in the form of nerve impulses, which can be considered as digital signals. However, the encoded information has the form of the pulse density, which is an analogue signal. Each nerve impulse arriving at the synapse generates an analogue internal potential in some proportion to the synaptic strength, which has either a positive or negative value corresponding to an exciting or inhibitory synapse (Figure 3.1b). These potentials are summed in a spatial-temporal way and when the total potential exceeds a value, called the threshold, a train of pulses is generated and travels along the axon.

3.3. The Artificial Model

3.3.1. Introduction

Some of the characteristics of biological neural networks discussed above can now be incorporated into an artificial neural network. Each neuron in the artificial network is connected to other neurons in the network, but not necessarily connected to the network input. Each of the connections between neurons has associated with it a synaptic weight. Each neuron computes some function of its specific inputs, perhaps as simple as a weighted sum or as complicated as a polynomial of several inputs, which then goes through a single weight. The computed neuron-input is then processed through a nonlinear function in the neuron to produce an output.

In mathematical terms a neural network model is defined as a directed graph with the following properties.

- A set of processing units (neurons, cells);
- A state variable (*state of activation*) n_i which is associated with each processing unit (neuron) *i;*

- A *weight* w_{ij} (synaptic strengths) associated with each link (synapses) *ij* between two neurons *i* and j, which determines the effect of the signal that unit j has on unit *i*;
- An *activation function* $f_i[n_j, w_{ij}, \vartheta_i, (j \neq i)]$ which determines the new state of each neuron *i* as a function of its activation threshold, of the weight of its incoming links and of the states of the other neurons connected to it by these links;
- An externally applied threshold-bias ϑ_i (*activation threshold*) for each neuron *i*, which has the effect of lowering (or increasing if it is negative) the network input of the activation function;
- A method for information gathering;
- An environment, within which the system must operate, providing input signals and-if necessary-error signals.

3.3.2. Processing units

Each unit performs a relatively simple job: receive input from neighbours or external sources and use this to compute an output signal, which is propagated to other units. Apart from this processing, a second task is the adjustment of the weights. The system is inherently parallel in the sense that many units can carry out their computations at the same time.

Within neural systems it is useful to distinguish three types of units: *input units* that receive data from outside the system, *output units* that send data out of the system and *hidden units* whose input and output signals remain within the system itself.

3.3.3. Connection between units

In most cases a valid assumption is that each unit provides an additive contribution to the input of the unit with which it is connected. By defining v_i as:

$$v_i = \sum_j w_{ij}(t) n_j(t) - \vartheta_i(t) \tag{3.1}$$

then v_i is called *activation potential* and the activation function is then usually taken to have the form:

$$f(v_i) = f\left(\sum_j w_{ij}(t) n_j(t) - \vartheta_i(t)\right) \tag{3.2}$$

Generally, some sort of threshold function is used for the activation function *f*, a hard limiter threshold function (sgn function), or a linear or semi-linear function, or a smoothly limiting threshold (Figure 3.2). The sigmoid (S-shaped) functions (nolinearities) like the following:

$$f(v_i) = \frac{1}{1 + e^{-v_i}} \tag{3.3}$$

or the hyperbolic tangent (tanh(i_i)) which yields output values in the range [-1,1], are more often used because they are continuous and differentiable.

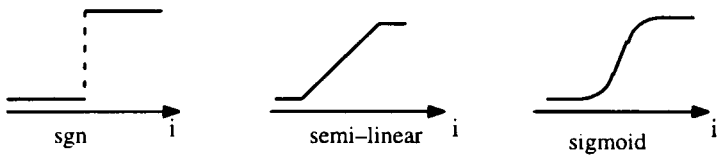


Figure 3.2 various activation functions for a unit

In Figure 3.3 the basic components of an artificial neural network, as described above, are given.

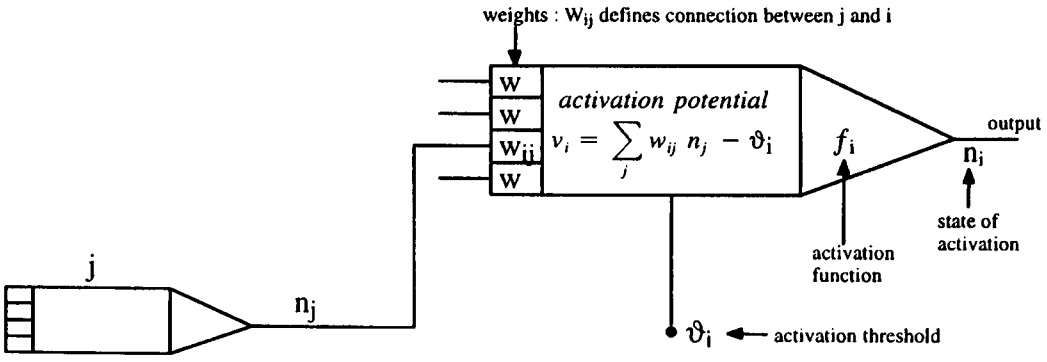


Figure 3.3 The basic components of an artificial neural network. The propagation rule used here is the standard weighted summation.

3.3.4. Network Topologies

Beside the processing units, different architectures can define an artificial neural network (Figure 3.4).

Feedforward Networks: The neurons of the output layer receive synaptic signals from those of the input layer but not vice versa and the neurons within one layer do not communicate with each other. The flow of information is thus strictly uni-directional. The synaptic weights are initially set to random values. The input array is multiplied by the layer of interconnection weights and summed by each of the neurons in the first layer. This weighted sum is processed through the activation function to compute the output of this first layer neurons and then multiplied again by the weights that connect them to the next layer of neurons. This process continues until the output neurons compute their outputs and finally a learning law is used to determine how to update the weights. Classical examples of feedforward neural networks are:

- *Radial Basis Function Networks.* The Radial Basis Function network (RBF) [48], [49] is a popular alternative to the Multilayer Perceptrons (MLP), which although it is not suited to larger applications as well as the Multilayer Perceptrons is, can offer advantages over the latter in some applications. The

only fundamental difference is the way in which hidden units combine values coming from preceding layers in the network. In a multilayer perceptron network, the hidden unit representations depend on weighted linear summations of the inputs, transformed by monotonic activation functions. In a Radial Basis Function network the hidden units use Euclidean distance to a prototype vector followed by transformation with a localised function.

Radial Basis Function network, in its most basic form, is a three-layer feedforward network. The input layer is made up of x source nodes. The second layer is a hidden layer consisting of j locally tuned units, which are fully interconnected to the output layer of l linear units. Each hidden unit output is obtained by calculating the "closeness" of the input x to an n-dimensional parameter vector associated with the jth hidden unit. The output layer supplies the response of the network to the activation patterns applied to the input layer. The transformation from the input space to the hidden-unit space is nonlinear, whereas the transformation from the hidden-unit space to the output space is linear.

In contrast with MLPs, which often have many layers of weights and a complex pattern of connectivity, RBF networks generally have a simple architecture consisting of two layers of weights, of which the first layer contains the parameters of the basis functions and the second layer forms linear combinations of the activations of the basis functions to generate the outputs. As a result, they provide very fast learning.

Like MLP networks, RBF networks are suitable for applications such as pattern classification, pattern recognition, interpolation, prediction and process modelling.

- *Time-Delay neural networks:* Time-Delay neural networks (TDNN) [50] can be considered as an extension of the static feedforward structure that is sensitive to temporal relationships in the input. The main feature of a time-delay neural network is the recognition of local features within a larger pattern, independent of the position of the local features in time. This is called time invariance.

Each feature neuron of a time frame is connected to a certain number of consecutive time frames (receptive field) of the next lower layer. For the next time frame of the layer, the receptive field of the lower layer is shifted by one but the same set of weights is used for the interconnection (weight sharing). If well designed, Time-Delay Neural Networks can have the same discriminative power as a large fully connected net but with a much smaller number of free parameters (independent weights). For the training of a TDNN, the regular backpropagation algorithm is applied to all time-shifted copies as if they were separate events. Then each weight is updated according to the average of all corresponding time-delayed weight changes. In this way, the network is forced to discover useful features in the input space, regardless of when in time they actually occurred.

- *Multilayer Perceptrons (MLP).* This is the most popular class of the feedforward neural networks. Multilayer Perceptrons [46] are used in this work so they are described in detail and their functions are explained in section 3.5.

Recurrent Networks: Such networks do contain feedback connections from output to input units. The inputs into this type of network are used as the initial conditions on the outputs of the neurons. These initial outputs are then multiplied by the fixed interconnection weights and summed at each neuron. Each connection has its own synaptic efficiency, *i.e.* weight, associated with it. Afterwards, this weighted sum is processed by the activation function to compute a new output for that neuron. These outputs are used as new inputs for the neurons they are connected to. The goal is for the network outputs to reach a stable state.

Unlike feedforward networks, recurrent networks can perform mappings that are functions of time and/or space or converge to one of a number of limit points. As a result, they are capable of performing more complex computations than feedforward networks. For example, they are capable of learning temporal patterns sequences, that is, sequences of patterns that are context or time dependent. Classical example of recurrent network is:

- *Kohonen's feature map*: Kohonen feature map [40], also called Self-organising map (SOM) is one of the best known artificial neural network algorithms. SOMs are a unique class of neural networks, since they construct topology-preserving mappings of the training data where the location of a unit carries semantic information. Therefore, the main application of this algorithm is clustering the data, obtaining a two-dimensional display of the input space that is easy to visualise.

  Self-Organising Maps consist of two layers of units: A one dimensional input layer and a two dimensional competitive layer, organised as a two dimensional grid of units. This layer can neither be called hidden nor output layer. Each unit in the competitive layer holds a weight (reference) vector that, after training, resembles a different input pattern. The learning algorithm for the SOM accomplishes two important things:

  1. Clustering the input data.

  2. Spatial ordering of the map so the similar input patterns tend to produce a response in units that are close to each other in the grid.

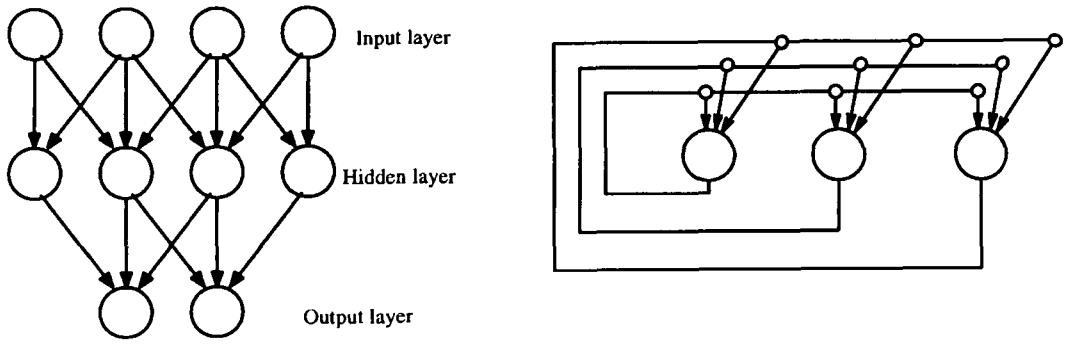


Figure 3.4 A feedforward network with one internal layer of neurons (**a**) and a recurrent network (**b**)

3.3.5. Training of artificial neural networks

Another way to categorise the different neural network models is by the basic classes of learning paradigms that are used for training. The three main classes of learning paradigms are:

- *Supervised learning or Associative learning* in which the network is adjusted, at each step, by comparing the actual output with the desired one which is presented simultaneously (Figure 3.5). Supervised learning is like trying to teach a new task to a child. After every attempt made to solve the problem, an attentive teacher will give specific, immediate, feedback to the child on how well they did. Supervised learning is a useful approach for training neural networks to perform classifications, functions approximation or models, and time-series forecasting, as load forecasting, where the network is trained to predict outputs at some point in the future. It is especially useful in problems where data in the form of input/output examples is available, but the exact transformation for processing the input and producing output is not known.

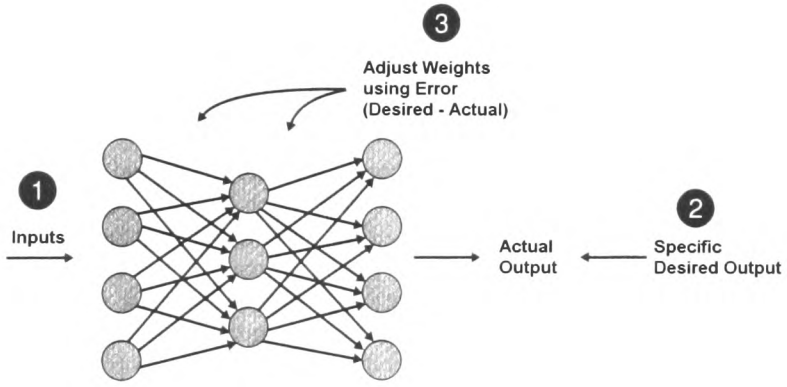


Figure 3.5 Supervised learning paradigm

- *Unsupervised learning or Self-organisation* in which the neural network adjusts its weights by self-organisation in response to input examples. Unsupervised learning is like being given a stack of documents, a filing cabinet with unmarked file folders, and having to create a coherent filing scheme from scratch. Unsupervised learning is used in cases where lots of data are

available, but the answer is unknown. The neural networks using unsupervised learning look at the data patterns and cluster them so that similar patterns get put into the same cluster (Figure 3.6). These neural networks are also called self-organising because they receive no direction on what the desired or correct output should be. When presented with a series of input patterns, the output processing units self-organise by initially competing to recognise the pattern, and then cooperating to adjust their connection weights. Over time an unsupervised network evolves so that each output unit is sensitive to, and will recognise, inputs from a specific portion of the input space.

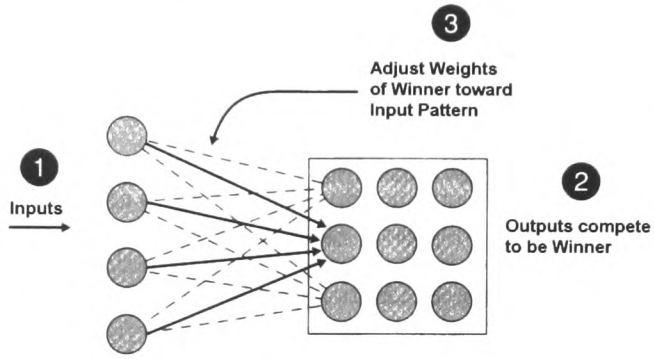


Figure 3.6 Unsupervised learning paradigm

- *Reinforcement learning* which is the on-line learning of an input-output mapping through a process of trial and error designed to maximise a scalar performance index called a reinforcement signal. In reinforcement learning, there are examples of the problem but the exact answer is not known, or at least not known immediately (Figure 3.7). For example. it is like playing a game, where moves are made by the opponents and after a number of turns the one wins and the other loses. This is the reinforcement signal. A series of decisions are made and only later are classified as wrong or correct. The reinforcement learning allows very difficult temporal (time-dependent) problems to be solved. When the problem involves some time sequential process or when the exact feedback is not available and only secondary signals are visible, the reinforcement learning is an appropriate technique to use.

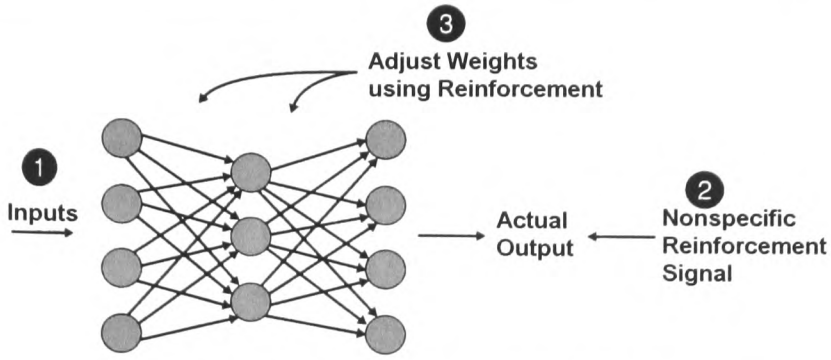


Figure 3.7 Reinforcement learning paradigm

3.4. Neural network models

The combination of connection topology, learning paradigm and learning algorithm define a neural network model. There is a wide selection of popular neural network models. While the most popular are multilayer perceptrons and Kohonen feature maps there are many other different types of neural networks in use. Some are optimised for fast training, others for fast recall of stored memories, and others for computing the best possible answer regardless of training or recall time. The best model for a given application depends on the nature of the problem and the type of the data available. The multilayer perceptron neural networks are used in this research to solve the forecasting problem, so they will be described and explained in detail in the next paragraphs.

3.5. Multilayer Perceptrons

*Multilayer perceptrons (*MLP) [41] are neural network architectures in which the input signal propagates in a forward direction (feedforward) and on a layer-by-layer basis. Their basic characteristics are:

- The network contains one or more layers of *hidden neurons,* which enable the network to learn complex tasks where simple perceptrons fail, such as non-

linear separable functions, by extracting more meaningful features from the input patterns.

- The network exhibits a high degree of *connectivity*, determined by the synapses of the network. This means that a neuron in any layer of the network is connected to all the neurons in the previous layer.
- Each neuron in the network includes a *smooth nonlinearity* (continuous and deferential everywhere) at its output end which is defined by the activation function. The presence of this nonlinearity is important because it distinguishes the layers of the multilayer perceptron and therefore enables the mapping between the input and the output variables to possess certain particular classification and discrimination properties.

Through the combination of these characteristics, together with the ability of such networks to learn from experience by training, the multilayer perceptrons derive their computational power in solving difficult and diverse problems, including load forecasting. However, the same characteristics are also responsible for the deficiencies in the knowledge regarding the behaviour of these networks. Indeed the non-linearity and the high connectivity together with the presence of the hidden layers makes the theoretical analysis and the visualisation of the learning process a difficult task.

It is now worth noting that in order to implement the neural network, the following parameters must be specified:

- *Number of inputs:* This must provide the information to the network in order to make a correct decision without being very large.
- *Connectivity of the network:* This issue involves the size of the network, that is, the number of the hidden layers and the number of neurons in each of these layers. There is not a standard rule that defines the size of a hidden layer. Tests have shown that if the hidden layers are large it will be difficult to train the network (too many parameters to estimate). On the other hand, if the hidden layers are too small the network may not be able to accurately classify

all the desired input patterns. Therefore, practical systems, such as the forecasting scenario, must balance these two competing effects.

In order to use a neural network efficiently, the values of the weighting coefficients and the thresholds for each computation element must be determined by the training procedure. Multilayer perceptrons are usually trained in a supervised manner with a highly popular algorithm known as the *error back-propagation.*

3.6. Error back-propagation algorithm

The central idea behind this algorithm is that the error signal which originates at the units of the output layer propagates backwards (layer-by-layer) through the network. Basically, the error back-propagation process consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an input vector is applied to the input nodes of the network, and its effect propagates though the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the network are all fixed. However, during the backward pass the synaptic weights are all adjusted in accordance with the error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backwards through the network, against the direction of synaptic connections. The synaptic weights are adjusted so as to make the actual response of the network move closer to the desired response.

The training algorithm can be realised via the following convergence steps:

- *Initialisation.* The first step is to initialise the network. In order to ensure that the network parameters (synaptic weights and threshold levels) are changed at all by the iterative convergencing learning procedure, the customary practice is to initialise them to small random numbers that are uniformly distributed inside a small range of values.

- *Presentation of the training examples.* A single data vector $\mathbf{z} = \{z_1, z_2, ..., z_k\}$ for the training examples is presented in the input, together with the respective desired output vector $\mathbf{d} = \{d_1, d_2, ..., d_m\}$ in the output layer (target vector).

- *Forward computation.* The activation potential for every neuron *j* in layer *l* of the network is computed from the equation :

$$v_j^{(l)}(t) = \sum_{i=1}^{p} w_{ji}^{(l)}(t)\, n_i^{(l-1)}(t) - \vartheta_j^{(l)}(t) \tag{3.4}$$

where $n_i^{(l-1)}(t)$ is the state of activation (output) of neuron *i* in the previous layer (*l-1*) ; $w_{ji}^{(l)}(t)$ is the synaptic weight which connects neuron *j* in layer *l* and neuron *i* in layer (*l-1*) and $\vartheta_j^{(l)}(t)$ is the threshold applied to neuron *j* in layer *l*, at iteration *t* ; *p* is the total number of inputs applied to neuron *j*.

Assuming the use of the logistic function of equation (3.3) for the sigmoidal nonlinearity, the output of each neuron *j* in layer *l* is then calculated:

$$v_j^{(l)}(t+1) = f(v_j(t)) = \frac{1}{1 + e^{-v_j^{(l)}(t)}} \tag{3.5}$$

When neuron *j* is in the first hidden layer (*l-1*), $n_i^{(0)}(t)$ is set equal to the *i*th element of the input vector $z_i(t)$.

By proceeding forward through the network, layer by layer, the state of activation (output) in the output layer $o_j(t)$ is found and the error signal can be finally computed as:

$$e_j(t) = d_j(t) - o_j(t) \tag{3.6}$$

Where $d_j(t)$ is the *j*th element of the desired response vector **d(t)**.

Thus the forward phase of the back-propagation algorithm begins at the first layer by presenting it with the input vector, and terminates at the output layer by computing the error signal for each neuron of this layer.

- *Backward computation.* The backward pass starts at the output layer by passing the error signals backward through the network, layer by layer, and recursively computing the local gradient δ for each neuron. Using the error signal, the local gradient δ for the neuron j in the output layer L can be computed as:

$$\delta_j^{(L)}(t) = e_j^{(l)}(t)o_j(t)[1-o_j(t)] \tag{3.7}$$

The local gradient for each neuron j in the hidden layer is determined recursively in terms of error signal of the units to which it directly connects and the weights of those connections:

$$\delta_j^{(l)}(t) = n_j^{(l)}(t)[1-n_j^{(l)}(t)]\sum_k \delta_k^{(l+1)}(t)w_{kj}^{(l+1)}(t) \tag{3.8}$$

Finally, the synaptic weights of the network in a layer are adjusted according to the generalised delta rule:

$$w_{ji}^{(l)}(t+1) = w_{ji}^{(l)}(t) + \alpha[w_{ji}^{(l)}(t-1)] + \eta\delta_j^{(l)}(t)n_i^{(l-1)}(t) \tag{3.9}$$

where η is the *learning rate parameter* and α is the *momentum parameter.*

- *Iteration.* Finally all the previous steps are repeated by presenting new training examples to the network until the free parameters of the network stabilise their values and the mean square error ε_{av} over the entire training is at a minimum or acceptably small value. The order of presenting the training examples must be randomised. The mean square error is given by the equation [41]:

$$\mathcal{E}_{av} = \frac{1}{N}\sum_{n=1}^{N}\sum_{j\in C} e_j^2(t) \tag{3.10}$$

where N denotes the total number of patterns, C includes all the neurons in the output layer of the network and e_j is the error signal of neuron j.

4. DATA PREPROCESSING

4.1. Introduction

As mentioned earlier (Section 1.3), the architecture of the complete load forecasting system consists of two basic modules. In this chapter the historical load data analysis is examined together with the preprocessing and the construction of the input set for the neural network. Data representation and preprocessing are extremely important to neural network applications. More than half of the development time is spent working with the data before it even sees the neural network. Thus having powerful data cleansing, and preprocessing operations are essential to effective results.

4.2. Data gathering

The historical load data were obtained from the Public Power Corporation and span four years, beginning from 1994. Hourly load raw data, along with two daily load peaks were available for the above time period. Daily minimum and maximum temperature measurements were also provided. Table 4.1 shows the raw records of the first week of June 1996 as taken from the Public Power Corporation. The structure of the record is described in Table 4.2.

96/06/01,Sa,133.8,120.2,112.6,108.4,106.4,105.1,116.3,147.3,179.3,200.2,214.0,224.8,223.6,206.9,182.6,171.3,172.7,185.4,193.2,204.9,247.6,230.9,196.1,161.7,227,250,21o,25o
96/06/02,Su,136.0,121.5,111.8,109.6,107.0,102.5,110.4,131.7,153.6,175.2,205.2,221.3,207.1,175.4,153.1,149.6,151.9,165.1,174.9,188.0,228.3,215.7,185.8,156.0,223,232,16o,26o
96/06/03,Mo,132.3,118.6,112.2,108.1,106.5,105.3,118.4,147.6,176.4,196.6,214.3,229.3,224.8,202.9,180.9,170.9,177.4,186.6,197.3,210.5,254.5,235.7,195.8,160.2,231,257,19o,26o
96/06/04,Tu,134.0,119.0,114.0,111.5,110.6,110.2,132.8,177.4,203.1,222.0,235.9,242.9,242.4,222.7,195.3,181.9,184.5,202.1,214.7,226.6,266.6,246.0,206.5,168.0,245,270,15o,28o
96/06/05,We,139.1,123.1,115.9,112.0,110.8,110.5,133.0,178.2,207.8,229.7,240.0,245.4,243.5,227.0,201.0,189.6,191.6,204.9,216.7,228.8,267.2,250.3,211.2,172.7,248,272,16o,29o
96/06/06,Th,143.8,129.2,121.5,116.8,115.2,115.2,139.4,183.2,215.0,237.9,248.1,256.4,251.7,230.1,201.1,190.4,192.6,210.3,221.9,232.1,273.6,255.5,217.4,178.2,256,278,20o,29o
96/06/07,Fr,146.2,131.6,125.1,119.7,118.0,118.1,137.1,182.7,218.8,236.6,248.1,256.4,257.1,235.6,208.9,196.0,195.9,213.7,222.2,231.3,270.8,249.7,213.1,175.0,260,274,18o,26o

Table 4.1 Raw data for the week 01/06/1996 - 07/06/1996. Each record consists of 24 load values of the hourly load demand, 2 load values for the daily and the night peak, the minimum an the maximum temperature.

Field Number	Field Description
1	Date
2	Day of the week
3-26	Hourly load demand from 00:00 to 23:00. (MW)
4	Noon peak load demand
5	Evening peak load demand
6	Minimum daily temperature
7	Maximum daily temperature

Table 4.2 Raw record description.

4.3. Data Analysis

Since multilayer neural networks can perform essentially arbitrary non-linear functional mappings between sets of variables, a single neural network could be used to map the raw data directly onto the required final output values. In practice, it is nearly always advantageous to apply preprocessing transformations to the input data before it is presented to a network. Similarly, the outputs of the network are often postprocessed to give the required values. These steps are indicated in Figure 4.1.

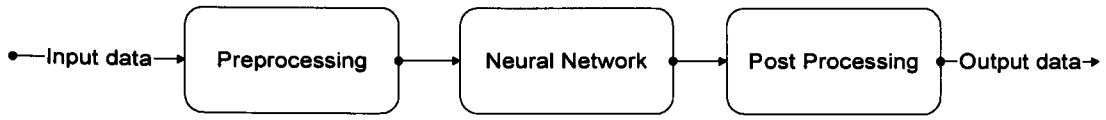


Figure 4.1 Illustration of the use of data pre-processing and post-processing in conjunction with a neural network mapping.

Data representation is important. If wrong decisions are made regarding representation, the neural network might not be able to learn the relationship that it is supposed to. However, there is usually a set of possible data representations that are sufficient to train a network. In all cases, it is important to understand how data representation decisions will affect both the training time for the neural network and the accuracy obtained.

In the simplest case, preprocessing may take the form of a linear transformation of the input data, and possibly also of the output data (postprocessing). More complex preprocessing may involve reduction of the dimensionality of the input data. At the simplest level this could involve discarding a subset of the original inputs. Other approaches involve forming linear or non-linear combinations of the original variables to generate inputs for the network. Such combinations of inputs are sometimes called *features,* and the process of generating them is called *feature extraction*. Depending always on the problem, dimensionality reduction can lead to improved performance, which may at first appear somewhat paradoxical, since it cannot increase the information context of the input data and in most cases will reduce it. However, it results in a network with fewer inputs, *i.e.* with fewer adaptive parameters to be determined, and these are more likely to be properly constrained by a data set of limited size, leading to a network with fewer weights that may be faster to train. In most of the situations though a reduction in the dimensionality of the input vector will result in loss of information. One of the main goals in designing a good preprocessing strategy is to ensure that as much of the relevant information as possible is retained. If too much information is lost in the preprocessing stage then the resulting reduction in performance more than offsets any improvement arising from a reduction in dimensionality.

Another important way in which network performance can be improved is through the incorporation of *prior knowledge*, which refers to relevant information which

might be used to develop a solution and which is additional to that provided by the training data. Prior knowledge can either be incorporated into the network structure itself or into the preprocessing and postprocessing stages.

A final aspect of data preparation arises from the fact that real data often suffer from a number of deficiencies such as missing and invalid input values or incorrect target values.

In general, one can note that it is necessary to examine the various characteristics and types of prior knowledge in order to represent it in a way that is both efficient and understandable to the neural network.

4.3.1. Data types

There are many data types in the historic data library, such as day of the week, load, temperature, hour *etc.* Most of these data can be easily mapped into three logical data types. These include continuous numeric values, discrete numeric values, and categorical or symbolic discrete values.

4.3.1.1. Numeric data representations

Numeric data can be simple binary values (0 or 1) indicating on/off states, or they can be a range of discrete values (1 to 10) or a continuous range from, say, -1000 to +1000. In each case a decision has to be made on how to scale and represent that data. Most neural networks accept inputs in a range of 0 to 1 or -1 to 1 depending on the activation function used in the neural network. In this case binary parameters can be represented by the extremes of the input range.

Discrete values: Discrete variables are ones that take on only a fixed set of values. These typically denote a small set of classes, a set of responses to multiple choice questions, or a fixed interval of integer values. The challenge for neural network representation of discrete values is to present these variable values in such a way that the network is able to discern the differences between

values and can tell the relative magnitude of the differences if that information is available. Various coded data types are used to represent these values. In the following paragraphs, the most commonly used codes are described.

One-of-N codes: When a variable can take on a value from a set of discrete values, it must be transformed into a representation that presents a unique set of inputs to the neural network for each distinct discrete value. The most common representation for discrete variables is the one-of-N code. A one-of-N code has a length equal to the number of discrete categories allowed for the variable, where every element in the code vector is a 0, except for the single element, which represents the code value. For example, if the day of the week is to be represented, Monday is represented as (1 0 0 0 0 0 0), Tuesday as (0 1 0 0 0 0 0), Wednesday as (0 0 1 0 0 0 0) *etc.* The advantages of the one-of-N code is that it is simple, easy to use, and the neural network can easily learn to discriminate between the various values. However, for variables with a large number of possible values, the one-of-N code can be very costly in terms of the size of the neural networks.

Binary codes: An alternative representation is the use of the standard binary code. Each discrete category is assigned a value from 1 to N and represented by a string of binary digits. That is, if 16 values are possible, they could be represented by a binary code vector of 4 digits. As long as the discrete values are arbitrary and not ordered in any way, a binary code is a fine way to represent data. However, note that there are large differences in the bit values as the discrete numbers get converted to binary codes. In the standard binary code, the eighth item is presented as (0 1 1 1), while the ninth has a code of (1 0 0 0). If the neural network should treat input patterns with neighbour values as "similar", then the Gray code might be chosen. The Gray code is used most often when the discrete values are related in some way, usually by increasing or decreasing values. Table 4.3 illustrates the Gray code for 16 values.

Decimal value	Gray Code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Table 4.3 Gray code

Continuous values: The most common form of data representation is scaling. A variable that can take on values from 0 to 1000 can be linearly scaled from 0 to 1. So 300 would take on a value of 0.3, while 700 would take on a value of 0.7. For evenly distributed variables like this, simple linear scaling works fine. If the data are skewed, for example, suppose 80% of the values are below 500, an option is to scale the data using a piece-wise linear approach. In this case, the data in the 0 to 500 range is expanded in the representation, while the less important 500 to 1000 is compressed. This can be done by taking the 0, 500, 1000 range and scaling that onto a 0, 0.8, 1.0 range. In this case an input of 500 would have a value of 0.8, while an input of 750 would be 0.9. A 250-count difference in the input value translates into only a 0.1 difference. Using this scaling, the neural network sees bigger differences in the input value and so can more easily discriminate between the differences in the input value.

4.3.1.2.Symbolic data representations

Not only numeric valued data are encountered as inputs in a neural network. Symbolic variables such as Boolean variables or variables such as the day of the

week often need to be represented as inputs in a neural network. The most common and easiest way to deal with this situation is to use one-of-N codes, where N is equal the number of valid values of the variable. The essential symbolic variables can also be represented as inputs to a neural network model and take discrete values in a range of [0,1]. For example, if Monday is represented as 0 and Sunday as 1 then the other days could be represented in the range of [0,1].

In general, the more explicit the data representation, the easier it will be for the neural network to learn. For example, taking a discrete variable and using a one-of-N vector code will typically train the fastest. However, the cost is that N input units and a factor of N additional weights are added to the network. Again, in general, the larger the network in terms of processing units and connection weights, the less well it will generalise and the longer it will take to train. Taking the same discrete variable and assigning it to a single input unit, where each discrete value is represented by a difference of 0.1 in input magnitude, is certainly a more compact representation. A smaller neural network and one that generalises better is likely to result. However, it will take the neural network a lot longer to adjust its weights from that single input unit in order to learn that a difference of a tenth is a significant difference that indicates a completely unique value for that input variable.

4.4. Rejection of faulty measurements

The data sets often contain inaccurate values, missing data, or other inconsistencies and they must go through a process known as "data cleansing" [42]. Every record must be checked and all the faulty and abnormal measurements must be rejected. Several techniques have been used to clean data. These include rule-based techniques, which evaluate each data item against the knowledge about the range of data expected in that data field and constraints or relationships to other data fields in every data record. Visualisation can also be used to easily identify outliers or out of range data in large data sets.

In order to apply the above-described techniques to the case of the power system of Crete various facts were taken into consideration. For example, in the island of Crete during the summer there is an abnormal increment of the load demand as a result of the increased population due to tourism. This excessive increase of the load demand during the summer often exceeds the available load demand, resulting in blackouts as indicated by minimum values in Figure 4.2.

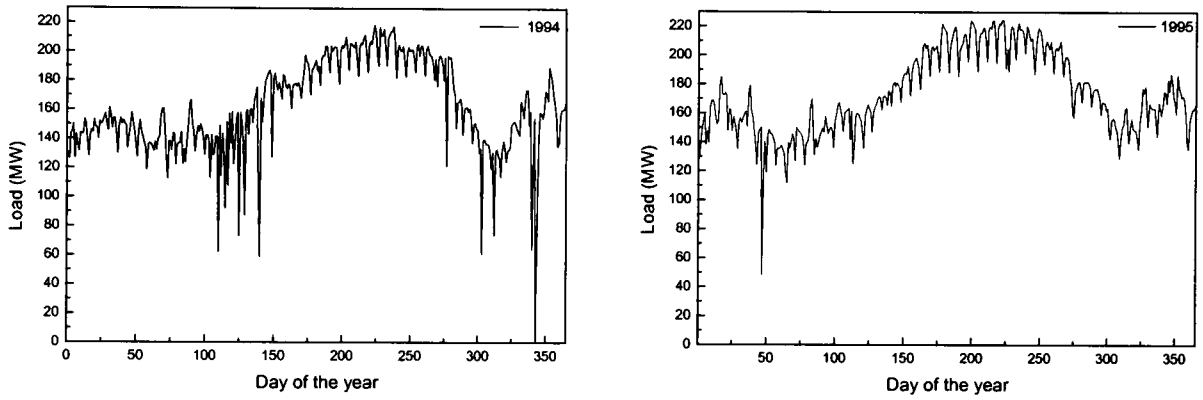


Figure 4.2 Average daily load demand for the years 1994 and 1995

In order to extract those faulty and abnormal measurements from the data library various rejection algorithms are used. Removing these measurements will not harm the generality, however it is believed that keeping them may lead to faulty prediction. Previous years' load demand data are used to calculate the typical load curve for each day in each season. Then a rejection process follows which consists of three stages:

a) *Construction of the typical load curves*: in this stage the typical hourly load for every day of the week (Monday to Sunday) and every season of the year (spring to winter) is produced. Thus a total of 28 (4 seasons by 7 days)

normalised load curves are produced. The typical normalised load curve of a day that belongs to a specific season (e.g. Monday of spring) is constructed as the average of the normalised load curves of the corresponding days of the previous years. The normalised load curve of a day is the transformation of that day's load curve to the [0,1] interval. The following relationship results in the previously mentioned transformation [24]:

$$\overline{L}(t) = \frac{L(t) - L_{\min}}{L_{\max} - L_{\min}} \tag{4.1}$$

where $\overline{L}(t)$ the normalised hourly load, $L(t)$ the hourly load, $L_{\min}$ and $L_{\max}$ the minimum and maximum values of the hourly loads of the day. The normalisation in the [0,1] space is taking place so that the factor of the yearly increase of the peak load will be removed while calculating the median.

b) *Rejection of faulty measurements:* The construction of the typical load curves is followed by a procedure that tracks down the faulty measurements. This procedure works as follows: the hourly load demand of each day of the selected year is compared to the corresponding typical and if their difference exceeds a threshold then this measurement is rejected. In the work described here the threshold was set to 15%.

c) *Graphical data processing: step* (b) is followed by a graphical representation of all measurements (including the ones rejected in the previous stage). This is intended to track any inconsistencies in the measurements and to facilitate the rejection of any faulty values that were not rejected during stage (b).

Figure 4.3 illustrates the average daily load demand for the years 1994 and 1995 after the rejection of the faulty and abnormal measurements.

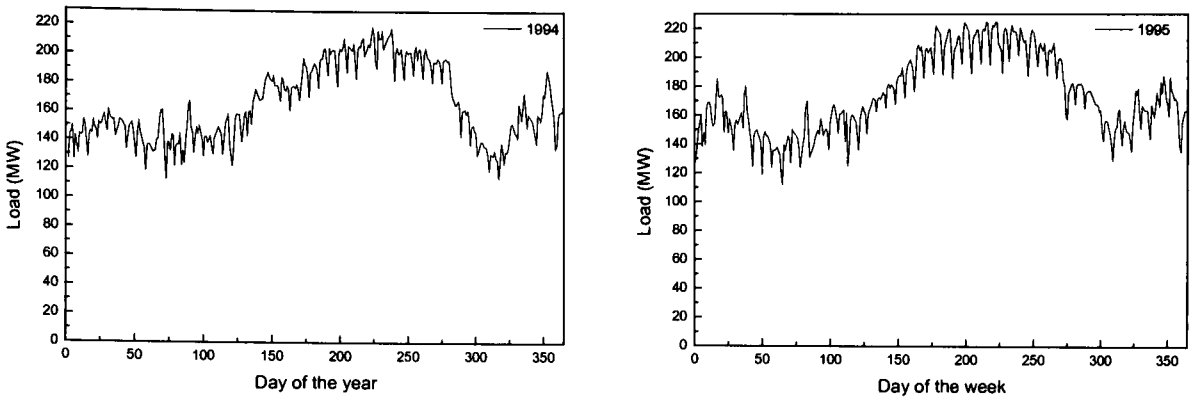


Figure 4.3 Average daily load demand for the years 1994 and 1995 after the rejection of the faulty and abnormal measurements

4.5. Input normalisation

The average annual load demand increases substantially every year due to the increasing development factor on the island caused by industry and tourism [3],[4]. In order to overcome this heterogeneous average annual load increment, a linear rescaling of the load values is applied to ensure that all the load values will be in the same value range. To do this, the monthly average load L_{av} is firstly calculated for every month and every year. Then a year was selected as a base year and all the raw load values L were normalised to monthly average L_{bav} of the base load year according to the following equation:

$$L_n = L \cdot \frac{L_{bav}}{L_{av}} \tag{4.2}$$

where L_n is the normalised load to the base year.

In experiments presented here 1994 was used as the base year, although the year selection is an arbitrary one and has no effects on the results.

Once the historical load data were normalised to the base year, all the data values are standardising to the [0,1] range. This rescaling is essential especially if various different inputs exist as in this case. If one input has a range of 0 to 1, while another input has a range of 0 to 500, then the contribution of the first input to the distance will be swamped by the second input. So it is essential to rescale the inputs so that their variability reflects their importance or at least is not in inverse relation to their importance.

The main emphasis in the neural network literature on initial values has been on the avoidance of saturation, hence the desire to use small random values. How small these random values should be depends on the scale of the inputs as well as the number of inputs and their correlation. Standardising inputs removes the problem of scale dependence of the initial weights. Figure 4.4 represents the normalised daily load demand for the years 1994 and 1995.

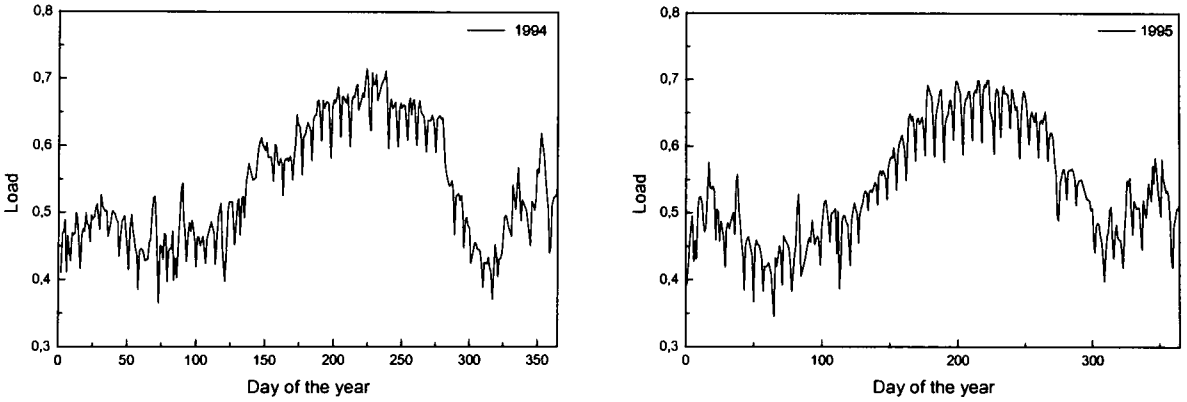


Figure 4.4 Average daily load demand for the years 1994 and 1995 after the normalisation to the base year and the rescaling to [0,1]

4.6. Input encoding

In many cases, transformations of symbols to other symbols are necessary before they are turned into numeric values. A common use would be to aggregate

members of some class or group into a single symbol for data representation purposes.

Symbolic to numeric translations are often required to turn discrete symbols or categories into numeric values for processing by the data preprocessing algorithms. The most basic form this can take is of a simple lookup table, where the symbol is compared against a list of symbols and when it is found, a corresponding numeric value is used.

The symbolic variables could be represented according to their order of appearance in the value set they belong to (e.g. the days of the week could have been represented as follows: Sunday – 0; Monday – 0.17; Tuesday – 0.34; ...; Saturday – 1). That way all nominal values are neither meaningful nor related with the load demand. So a special representation for the symbolic variables is introduced in this research work. The symbolic variables are represented according to their functional relationship with the average load demand. The following stages are followed in order to achieve this representation:

a) Calculating the mean value of the annual load demand for every value of the symbolic variable. Specifically, the mean yearly load demand was calculated for:

    - every day of the week,
    - every hour of the day,
    - every month of the year,
    - every season of the year and
    - every temperature.

b) Then the symbolic variable with the largest mean yearly load demand value was represented by 1 and the one with the smallest with 0. The rest of the values were represented by a value in the (0, 1) interval respectively. Thus

Sunday (that is the day with the smallest mean yearly load demand) is represented by 0 while Thursday is represented by 1.

This non-linear mapping of the nominal value day of the week to the resulting input is shown in Figure 4.5, while the non-linear mapping of the nominal value hour to the resulting input is shown respectively in Figure 4.6. Figures 4.7, 4.8 and 4.9 illustrate the non-linear mapping of the month, the minimum temperature, and the maximum temperature respectively.

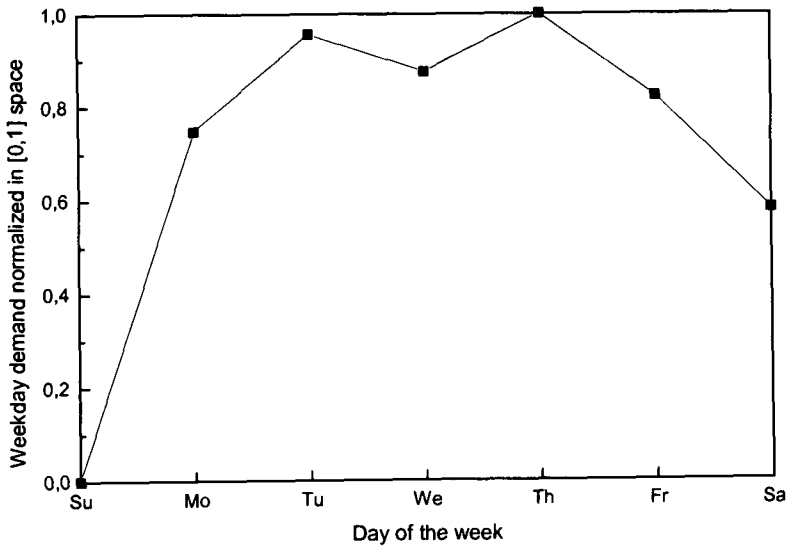


Figure 4.5 Weekday demand normalised in [0,1] space

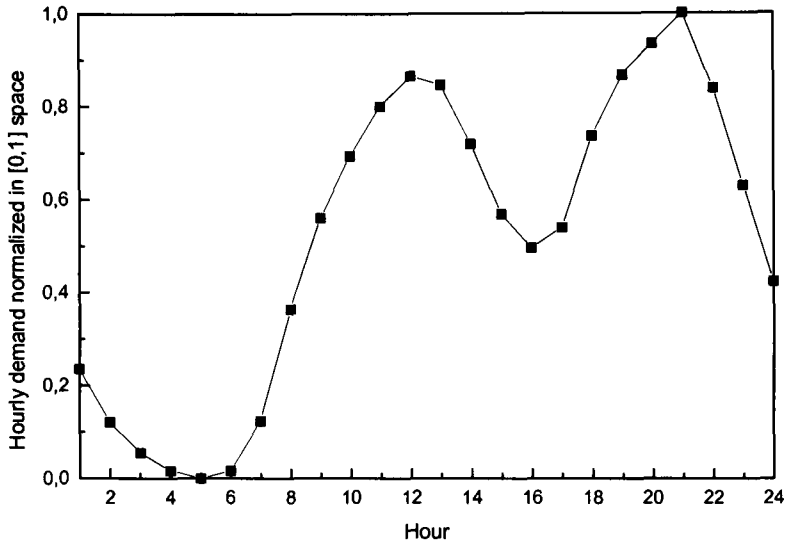


Figure 4.6 Hourly demand normalised in [0,1] space

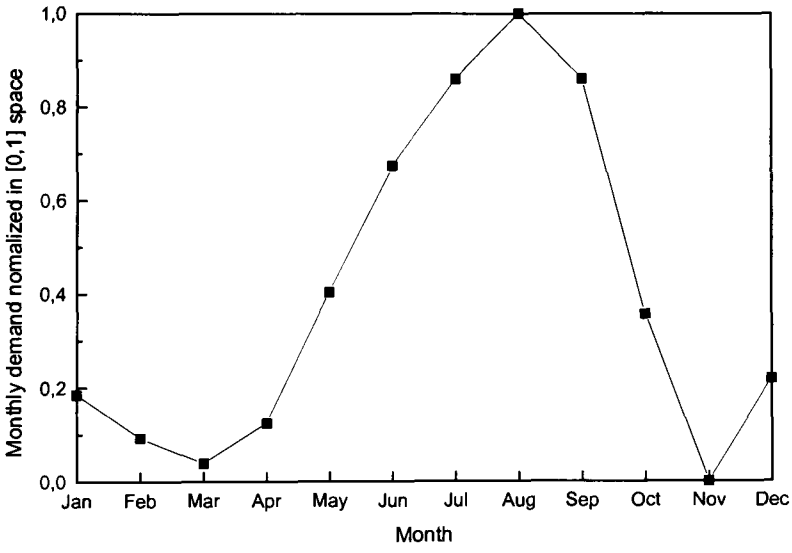


Figure 4.7 Monthly demand normalised in [0,1] space

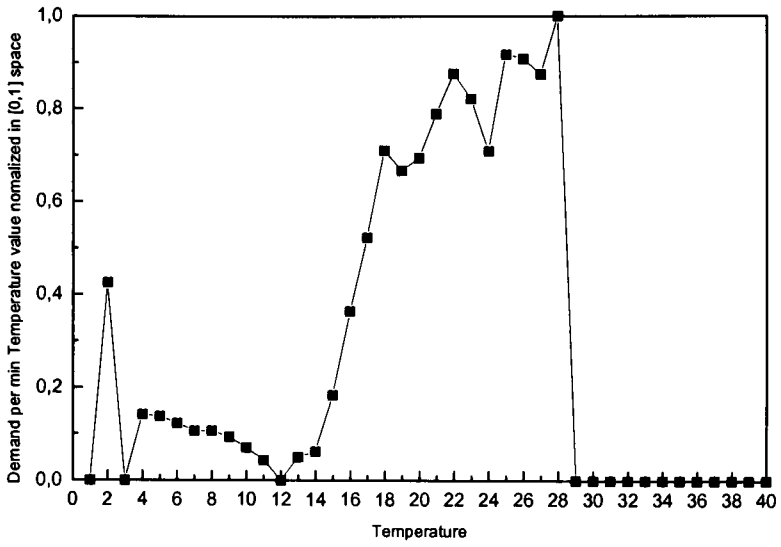


Figure 4.8 Demand per minimum Temperature value normalised in [0,1] space

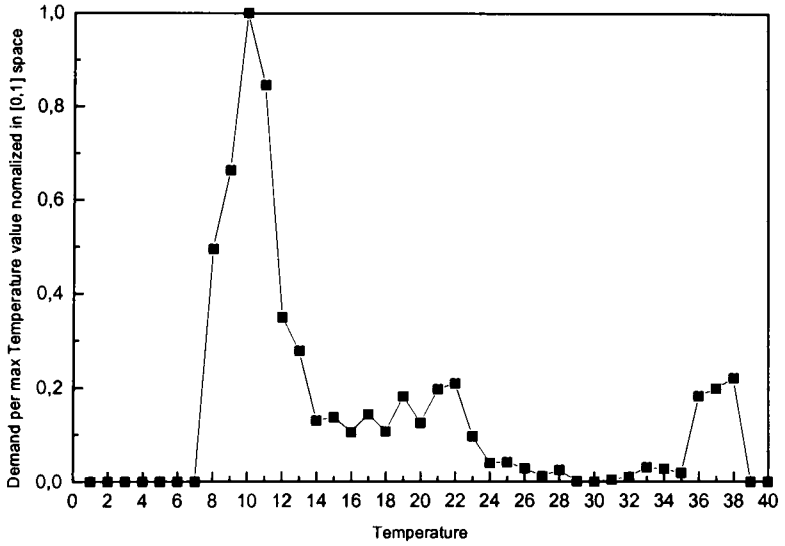


Figure 4.9 Demand per maximum Temperature value normalised in [0,1] space

4.7. Output representation

In most Neural Network applications for load forecasting, load values take the form of continuous variables and are presented to the network as a single neuron indicating the exact prediction value. In this research work a different and unique approach is introduced: the output load value is stored as an ordinal variable indicating various increments of load prediction. Using this kind of output representation, instead of predicting the exact load demand, a small interval that will contain the actual value is predicted. The prediction interval was set to 10MW, which was considered as reasonable, keeping into consideration the fact that power generators do not produce loads less than 10 MW. The above decisions were made after analysing the needs of the production plant, which was performed jointly with the Public Power Corporation management. This is reasonable since what is discussed here is prediction and it is unsafe and unnecessary to try and predict the exact value.

Sixteen intervals of load demand have to be represented as output to the neural network. Discrete Value Representation is then used as described in Section 4.3.1.1. The representation used is:

a) One single neuron representation. The 16 intervals are represented as values from 0 to 1 (Table 4.4). For example, the output value [0.25, 0.3125] corresponds to load demand [135, 145] MW.

b) One-to-N representation. 16 "dummy" binary variables are used for discriminating among the 16 different load ranges (Table 4.4). Each bit is given value 0 except the one corresponding to the correct load range, which is given a value of 1.

c) Binary code representation. This was the representation that was more highly valued for the representation of the output neuron. A special flavour of binary representation, the Gray code, is introduced and was used throughout this research work. The basic advantage that the Gray code representation has is that the binary numbers change only one bit each time they are incremented

by one. For example, 7 is represented by 0110 and 8 by 1100. Three (the 3 last ones) out of 4 bits remain the same and only the first one changes from 0 to 1 (Table 4.4). Having the neighbouring intervals being represented by numbers differing by only one bit is a very important factor for convergence. Furthermore, the Gray code representation provides error tolerance that cannot be found when using the standard binary representation or the one-to-N representation. This can be explained be looking at the probability of an error in the output by one bit (for example somehow the output gets confused and one bit changes value). In two of the four possible changes (probability 50%) using the Gray code will move the result to the next level (higher or lower) and will induce a non-fatal error unlike the one-to-N representation or to the lesser extent the standard binary representation. The experimental results verify this as shown in section 5.7.4.1.

Load interval (MW)	Equivalent Range	Single neuron representation	one-of-16 representation	Gray Code representation
95-105	0	0 - 0.0625	1000000000000000	0000
105-115	1	0.0625 - 0.125	0100000000000000	0001
115-125	2	0.125 - 0.1875	0010000000000000	0011
125-135	3	0.1875 - 0.25	0001000000000000	0010
135-145	4	0.25 - 0.3125	0000100000000000	0110
145-155	5	0.3125 - 0.375	0000010000000000	0111
155-165	6	0.375 - 0.4375	0000001000000000	0101
165-175	7	0.4375 - 0.5	0000000100000000	0100
175-185	8	0.5 - 0.5625	0000000010000000	1100
185-195	9	0.5625 - 0.625	0000000001000000	1101
195-205	10	0.625 - 0.6875	0000000000100000	1111
205-215	11	0.6875 - 0.75	0000000000010000	1110
215-225	12	0.75 - 0.8125	0000000000001000	1010
225-235	13	0.8125 - 0.875	0000000000000100	1011
235-245	14	0.875 - 0.9375	0000000000000010	1001
245-255	15	0.9375 - 1	0000000000000001	1000

Table 4.4 Various representations of the load ranges

5. EXPERIMENTS AND RESULTS

5.1. Introduction

The material presented in the previous chapter describes all the preprocessing and postprocessing procedures followed to produce the necessary historical database for a load forecasting system. The type and the architecture of the artificial neural networks used together with the experiments and the results will be examined in this chapter.

5.2. Network parameters

Table 5.1 lists the variable parameters of a multilayer perceptron neural network trained with the error back-propagation algorithm.

Parameter	Symbol	Typical Range
Error tolerance	T	[0,0.1]
Learning rate	η	[0,1]
Momentum	α	[0,1]

Table 5.1 Variable parameters of multilayer perceptron neural networks

Learning rate coefficient. The effectiveness and convergence of the error back-propagation learning algorithm depends mainly on the value of the learning constant η $(0 < \eta < 1)$. On the one hand, a small value of η will cause small changes to the synaptic weights in the network from one iteration to the next and also a very smooth trajectory in the weights through learning. However, the cost of this is an increased total number of training cycles that need to be made in order

to reach a satisfactory convergence. If, on the other hand, a large learning rate η is used, the learning speed will increase but the large changes in the synaptic weights may drive the network into an unstable position (e.g. oscillation).

Momentum coefficient. The momentum term serves to smooth and accelerate the convergence process and also to avoid the danger of instability during the learning. Its effect is to filter out high-frequency changes in the weight values, so that there is less chance that the neural network will start oscillating around a set of values. The momentum parameter causes the errors from previous training patterns to be averaged together over time and added to the current error. So if the error on a single pattern forces a large change in the direction of the neural network weights, this effect can be mitigated by averaging the errors from the previous training patterns. This is especially efficient if the previous pattern errors were forcing the network weights in the opposite direction. Instead of using error information from a single training pattern (as would be the case when momentum is set to 0), the errors from the prior patterns are averaged in. The overall result is that the weights are less likely to be driven back and forth in alternate directions.

Error tolerance. The error tolerance parameter specifies how close the output value must be to the desired value before the error is considered to be zero. In many cases, an error tolerance of 0.1 is used. This means that if the target value is 1.0, a network output value above 0.9 (1.0 - 0.1 =0.9) is within the tolerance, and the error is treated as 0.0. One of the main reasons for using an error tolerance is to avoid driving the network weights to extreme values. If the output unit activation values are kept in the range of 0.1 to 0.9 (with a tolerance of 0.1), the logistic function remains linear. As the output values climb up to 1.0 or down to 0.0, the sum of input signals must be quite large. Since the outputs of the other units will only be in the range of 0.1, this usually requires that the weights grow larger. Once the weights grow to large values and the output of the logistic function is above the knee of the S-shape curve, it is quite hard to change the output of the unit. This condition is called "network paralysis" [43].

What are the optimal values of the above parameters? Most work on feedforward neural networks uses constant values of these parameters. Rumelhart [41] recommended that a combination of learning rate η = 0.25 and a momentum parameter α = 0.9 can yield good results for most problems. But there is still no consensus as to what values of η and α should be used in the learning process. In fact, the optimal values η and α may be problem dependent. In this research work, extensive studies were performed on the effect of different values of η and α on the convergence rate and the performance of the system.

5.3. Network result evaluation

To measure the network's efficiency, the definition of the following terms is required:

Performance: To evaluate the resulting neural network's performance, the following absolute average percentage error measure is utilised:

$$\%Error = \frac{abs(actual - forecasted)}{actual} * 100 \qquad (5.1)$$

Network performance is determined as the difference of the *%Error* from the 100%.

$$\%Performance = 100 - \%Error \qquad (5.2)$$

Absolute difference from the actual: For a load demand forecasting system equally important to the performance factor is the difference of the faulty measurements from the actual value. The magnitude of this difference if too high can lead to substantially faulty results. So to ensure a complete evaluation of the systems that will be analysed along with the performance, the percentage of the

faulty predicted load values as a function of their difference from the actual values will be calculated.

5.4. Training infections

5.4.1. Underfitting-Overfitting

The critical issue in developing a neural network is generalisation: how well will the network make predictions for cases that are not in the training set? Neural networks can suffer from either underfitting or overfitting. A network that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to underfitting. A network that is too complex may fit the noise, not just the signal, leading to overfitting. Overfitting is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data. Overfitting can also produce wild predictions.

The best way to avoid overfitting is to use lots of training data. If at least 30 times as many training cases as there are weights in the network are available, it is unlikely to suffer from much overfitting, although it may get some slight overfitting no matter how large the training set is. For noise-free data, 5 times as many cases as weights may be sufficient. But in general we can not arbitrarily reduce the number of weights for fear of underfitting.

5.4.2. Overtraining

When training a neural network, it is important to understand when to stop. It is natural to think that if a network is trained for 100 iterations (epochs) is good, then 1000 epochs will be much better. However, this intuitive idea of "more practice is better" does not hold with neural networks. If the same training patterns or examples are given to the neural network over and over, and the weights are adjusted to match the desired outputs, the network is essentially instructed to memorise the patterns, rather than to extract the essence of the relationships.

What happens is that the neural network performs extremely well on the training data. However, when it is presented with patterns that are not part of the training patterns, it cannot generalise and does not perform well. This problem is called overtraining. On the other hand, if the training stops too early, the network is not capable of performing the task it was trained for. It is necessary therefore to balance carefully between these two effects, thus a convergence criterion is utilised to terminate the learning procedure. An effective convergence criterion deals with the best generalisation ability (performance) of the network. This means that the learning procedure stops as soon as the performance reaches its maximum value. It was therefore decided to set an error limit that varies from 0.2 to 0.3 depending on the neural network used. The error is defined as the sum of the squared difference between each of the calculated outputs and the corresponding desired outputs (targets) during one training cycle as given in equation (3.10). After the above error limit is reached the performance of the network is examined every ten epochs and the training procedure is finished as soon as the performance reaches its maximum value.

5.5. Experimental conditions

5.5.1. The historical database

To provide some coherence to the results presented, it was decided to use the same training database for every network trained with different parameters. Therefore, the results can be compared.

Training set: A set of examples used for learning that is to fit the parameters (weights) of the classifier. It is important to have enough data to yield sufficient training and test sets to train and evaluate the performance of neural networks efficiently. The amount of data required for training a network is heavily dependent on the network architecture, the training method, and the problem being addressed. In most of the experiments a training database of one year's historical data is used. Extensive studies were also performed on the effect of the number

of training patterns on the network performance. The results are illustrated in the following subsections.

Test set: A set of examples used only to assess the performance (generalisation) of a fully-specified classifier. All the available historical data that are not part of the training database were used as test sets in this work.

The database sets created and used in the research are described in Table 5.2.

Database name / *Type*	Description	Input neurons	Output neurons	Patterns
HTDB1 *Training*	01/01/1994 - 31/12/1994	20	4 bit encoded	7344
HTDB2 *Training*	01/01/1994 - 31/12/1994 Randomly selected patterns	20	4 bit encoded	3520
HTDB3 *Training*	01/01/1994 - 31/12/1994 Randomly selected patterns	20	4 bit encoded	592
HUDB1 *Test*	01/01/1995 - 31/06/1996	20	4 bit encoded	10152
HTDB4 *Training*	01/01/1994-31/12/1994	20	16 one-of-n	7344
HUDB2 *Test*	01/01/1995 - 31/06/1996	20	16 one-of-n	10152
DTDB1 *Training*	01/01/1994-31/12/1994 Whole day prediction	61	24	320
DUDB *Test*	01/01/1995-31/06/1996 Whole day prediction	61	24	534
NPTDB1 *Training*	01/01/1994 - 31/12/1994 Noon load peak prediction	10	4 bit encoded	159
NPUDB *Test*	01/01/1995-31/06/1996 Noon load peak prediction	10	4 bit encoded	448
EPTDB1 *Training*	01/01/1994-31/12/1994 Evening load peak prediction	10	4 bit encoded	159
EPUDB *Test*	01/01/1994-31/12/1994 Evening load peak prediction	10	4 bit encoded	448

Table 5.2 Available database sets

Symbol	Description
H	Hourly
D	Daily
NP	Noon peak
EP	Evening peak
T	Training
U	Untrained
DB	Database set

Table 5.3 Explanation of Table 5.2 symbols

5.5.2. The neural network simulator used

The experiments were implemented with the use of SNNS [47], an artificial neural network simulation environment developed at the Institute for Parallel and Distributed High Performance Systems at the University of Stuttgart. Details of the SNNS simulator are given in Appendix B.

5.6. Network training

The networks' training was based on repeated presentations of the complete set of the training patterns (training examples and target vectors together) from the training database. One complete presentation of the entire training set is called an *epoch*. The order of presentation of the training examples from epoch to epoch is taken randomly. The weight updating was performed after the presentation of each training example. This randomisation and pattern-by-pattern updating of the weights made the learning process *stochastic* in nature, which in turn drastically reduced both the learning time in classification tasks and the number of local minima.

The learning process was maintained until the network parameters (weights and threshold levels) were stabilised and the average square error over the entire training set went to zero. However, the target values 0 and 1 can never be achieved by the output of such a system due to the sigmoidal activation function. This function tend to 0 and to 1 only when the activation potential tends to ($-\infty$) and ($+\infty$) respectively (see Equation 3.3). Therefore the target values are modified and set to 0.1 for the "low" value and 0.95 for the "high" value.

5.7. Hourly load demand prediction

5.7.1. The network architecture

Initially a 20 input neuron network is selected for the prediction of the next hour load demand. The training patterns are extracted from the preprocessed historical database and described in Table (5.4). Two hidden layers were selected with 20 hidden neurons each and four output neurons for the Gray code representation of the load range to be forecasted. The network architecture used is presented in a graphical form in Figure 5.1.

NODE	DESCRIPTION
1-4	The load demand last four hours before the prediction
5-8	The load demand of the previous day before the prediction day as follows: the same hour, previous two hours, previous three hours of the forecast hour
9-12	The load demand of the previous week before the prediction day as follows: the same hour, previous two hours, previous three hours of the forecast hour
13-14	Minimum and maximum temperature values one day before the prediction day
15-16	Minimum and maximum temperature values one week before the prediction day
17-18	Minimum and maximum forecast temperature values at the prediction day
19	The day of the week
20	The hour to be forecasted

Table 5.4 Summary of the input variables used to the network for the hourly load prediction

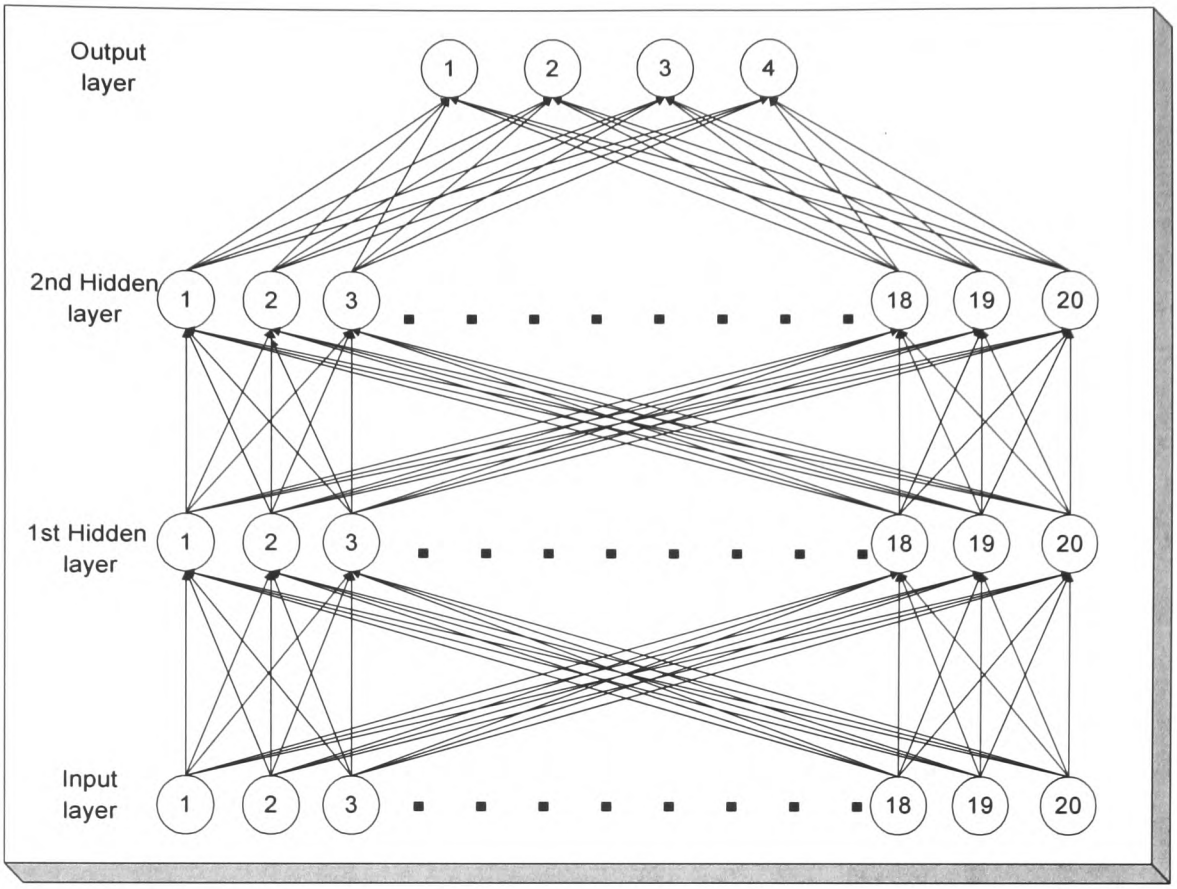


Figure 5.1 Architectural graph of the basic neural network for the hourly load prediction

5.7.2. The effect of the learning rate and momentum parameter

5.7.2.1. Learning rate equals 0.3, momentum parameter equals 0.9

Initially, a learning rate of 0.3 and a momentum parameter of 0.9 were selected. The performance reached its maximum value at the 125^{th} epoch. The mean square error using the training sets at that point was *MSE*=0.17.

The mean square error as a function of the learning iterations is illustrated in Figure 5.2 while the percentage of the error of the network using the untrained database as a function of the epochs if given in Figure 5.3.

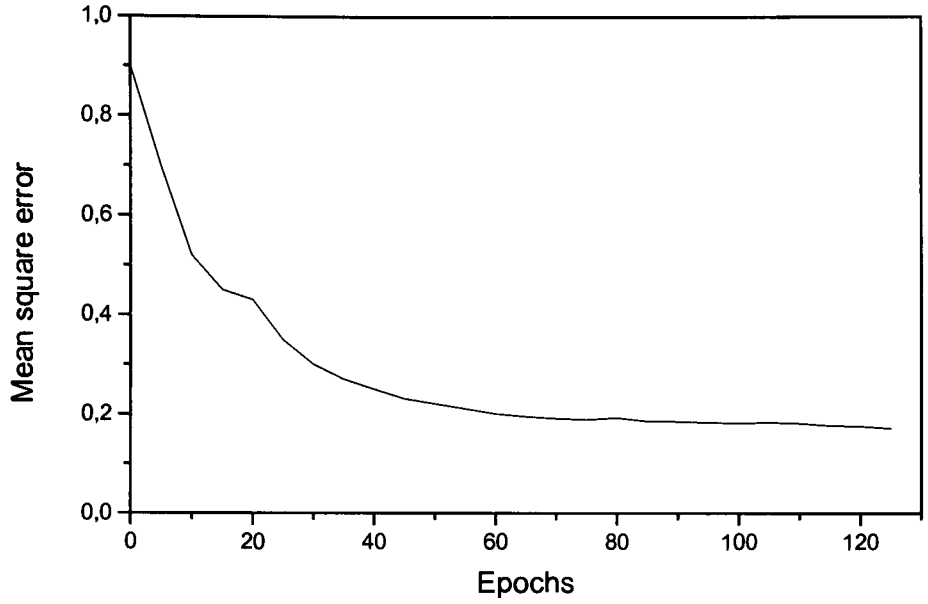


Figure 5.2 Mean square error as a function of the learning epochs

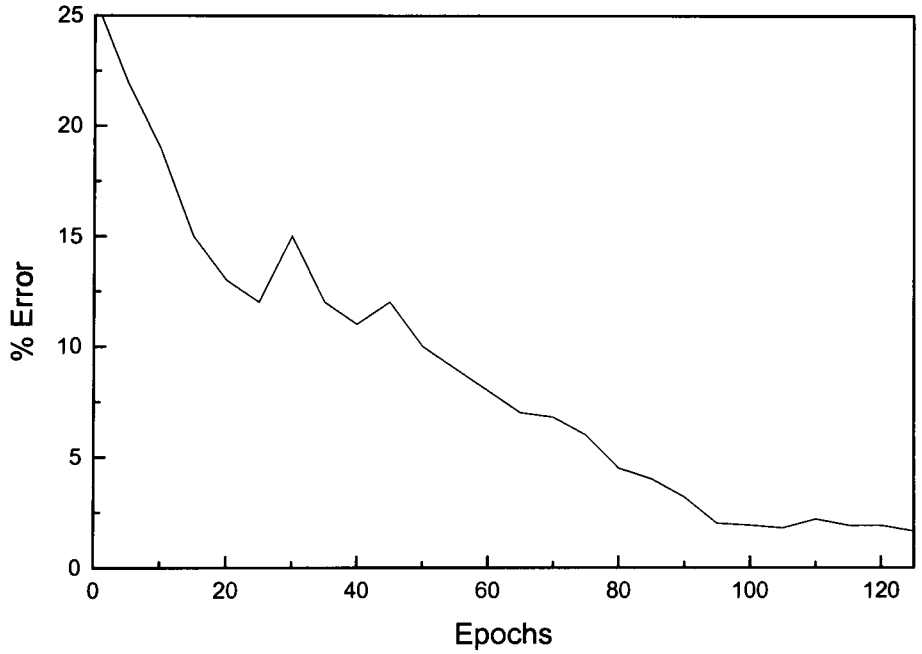


Figure 5.3 Percentage of the error of the network as a function of the learning epochs

Table 5.5 illustrates the sixteen load ranges used for the prediction while Table 5.6 summarises firstly the results after the test procedure using the untrained database and secondly the percentage of the faulty forecasted load values, depending on their difference from the actual values. As shown in Table 5.6 the majority of the faulty forecasted load values are very close to the actual. Figure 5.4 illustrates the actual hourly load values of a typical day (red line) together with the predicted hourly load range (grey area).

Load range	Load interval (MW)	Load range	Load interval (MW)
0	95-105	*8*	175-185
1	105-115	*9*	185-195
2	115-125	*10*	195-205
3	125-135	*11*	205-215
4	135-145	*12*	215-225
5	145-155	*13*	225-235
6	155-165	*15*	235-145
7	165-175	*15*	245-255

Table 5.5 Load ranges used for the prediction

Learning rate η	**Momentum α**	**Number of epochs**	**Mean square error (MSE)**	**%Error**	**%Correct**
0.3	0.9	125	0.17	1.65	98.35

|actual load - forecasted load|	**10 MW**	**20 MW**	**30 MW**	**40 MW**	**50 MW**
% of the faulty predicted loads, depending on their difference from the actual	95.30	3.70	0.76	0.16	0.08

Table 5.6 Neural network results for learning rate equals to 0.3 and momentum parameter equals 0.9

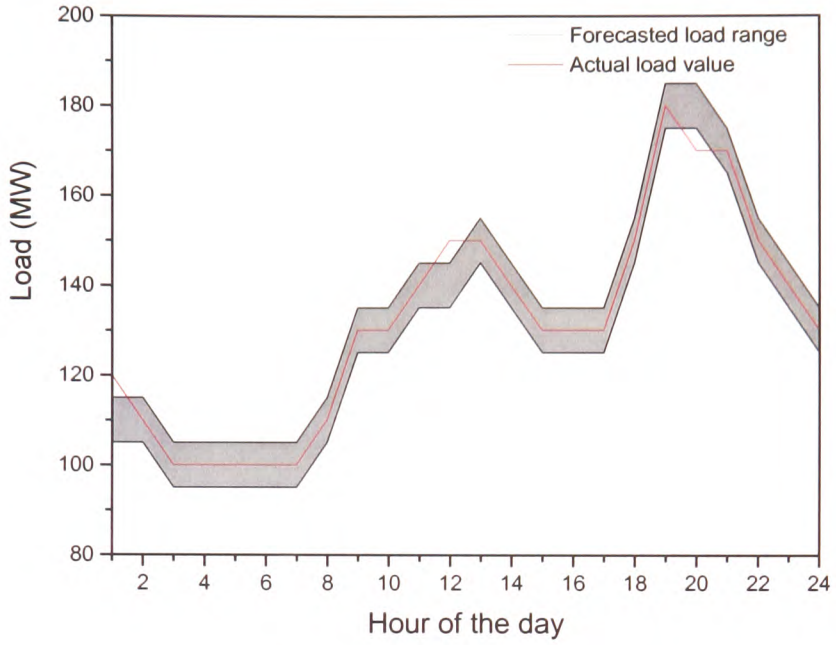


Figure 5.4 The actual load curve and the forecasted load range of a typical day

5.7.2.2. Learning rate equals 0.6, momentum parameter equals 0.8

A learning rate of 0.6 and a momentum parameter of 0.8 were also examined. Although, the performance of the network did not change significantly, a slight increase in the percentage of the faulty predicted load values was observed which exceeded the larger differences from the actual by a factor of 20MW or greater.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.6	0.8	101	0.19	1.86	98.14

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	94.60	3.60	1.26	0.54	0

Table 5.7 Neural network results for learning rate equals to 0.6 and momentum parameter equals 0.8

The conclusion that can be extracted from the experiments described above is that the most effective network is the one trained with a learning rate of 0.3 and a momentum term of 0.9.

5.7.2.3. Adaptive learning rate and momentum parameter

These parameters may also be adjusted during the training phase in a way that permits a better convergence. A variable setting of the parameters, that combines the advantages of the constant settings described above, was used. The learning rate coefficient was kept high (η=0.6) and the momentum term was kept low (α=0.5) during the first learning phase (until a certain error limit of 0.35) to allow large weight modifications and – hopefully - settle into an approximate mapping as quickly as possible. To achieve a stable coarse representation, smaller changes must be made to the weight vectors at each node. Therefore, the learning rate was decreased by 0.02, after every epoch, until it reached the value of 0.1 and simultaneously the momentum was increased by 0.02 until a maximum value of 0.92, both after every epoch. When the learning rate reached its minimum value, it was incrementally decreased further more by 0.005 until a minimum value of 0.01 was reached, whereas the momentum term remained constant.

The results using the above adaptive learning algorithm are described in Table 5.8.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
Adaptive	adaptive	149	0.16	1.68	98.32

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	95.11	3.87	0.74	0.16	0.12

Table 5.8 Neural network results for adaptive learning rate momentum parameter

The performance in the case with the constant setting of the learning rate parameter equal 0.3 and momentum parameter equal 0.9 reaches a higher value compared with the performance using the adaptive learning parameters. Although

the MSE in the case with the adaptive learning reaches a minimum value, the network seems to overlearn the training examples (Table 5.9).

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	125	0.17	1.65	98.35
0.6	0.8	101	0.19	1.86	98.14
adaptive	adaptive	149	0.16	1.68	98.32

|actual load – forecasted load|		10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	η=0.3,α=0.9	95.30	3.70	0.76	0.16	0.08
	η=0.6,α=0.8	94.60	3.60	1.26	0.54	0
	η, α adaptive	95.11	3.87	0.74	0.16	0.12

Table 5.9 Comparative table with the results using various learning rate and momentum parameter.

5.7.3. The effect of the number of the training patterns

5.7.3.1. A reduction of the training patterns

A reduction of the training patterns had a minimal effect on the performance. Various experiments were performed using a training database with fewer patterns (HTDB2). The results are illustrated in Table 5.10.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	198	0.18	1.87	98.13

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	94.61	4.31	0.86	0.11	0.11

Table 5.10 Neural network results with the use of the diminished training database (HTDB2)

5.7.3.2. A further reduction of the training patterns

A further reduction in the number of training patterns to 592 (HTDB3) caused a substantial increase in the forecast error as shown in Table 5.11. Even though the mean square error reaches a new minimum value, the network is unable to generalise with such a small training database.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	295	0.12	5.56	94.44

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	64.46	26.67	7.35	1.23	0.29

Table 5.11 Neural network results with the use of the fewest training database (HTDB3)

5.7.4. The effect of the output representation

5.7.4.1.A neural network architecture without the bit encoded output

A neural network with sixteen output nodes was also examined. Every single output neuron represents a load range and all the neuron are zero except for the neuron which represents the output load range. The network was trained with the use of the (HTDB1) training database. Although the above network topology had poor effects on the network efficiency as illustrated in Table 5.12, there is an increase in the percentage of the faulty measurements that exhibits differences from the actual, by a factor of 20MW or greater (Table 5.12).

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	310	0.32	2.01	97.99

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	81.27	12.58	4.49	1.44	0.22

Table 5.12 Neural network results with the use of 16 output nodes (2 hidden layers, 20 nodes each)

5.7.4.2. The effect of the number of the hidden neurons to a 16-output neural network

Another network used had the same input and output neurons as the above but with 30 hidden neurons. The efficiency of the network was slightly better compared with the one with 20 hidden neurons as illustrated in Table 5.13, but an increase in the training time was noted.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	413	0.34	1.96	98.04

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	85.68	8.83	4.32	1.06	0.11

Table 5.13 Neural network results with the use of 16 output nodes (2 hidden layers, 30 nodes each)

5.7.4.3. The effect of the learning rate and the momentum parameters to the 16-output neural network.

The same neural network but with the use of an increased learning rate η equals 0.5 and a momentum parameter equals 0.8, was also examined. The

performance of the network using the untrained database was slightly diminished as shown in Table 5.14.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.5	0.8	345	0.36	2.23	97.77

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	87.37	7.48	3.97	0.95	0.23

Table 5.14 Neural network results with the use of 16 output nodes and an increased learning rate parameter.

Table 5.15 summarises the performance results using the 4 output and the 16 output network topologies described above while Figure 5.5 illustrates the average difference of the predicted load value from the actual as a function of the load range for both 4 and 16 output neurons. Although the effect on the network performance using the untrained database is minimal, there is a substantial increase on the percentage of the faulty load demand values that exhibit the larger differences from the actual, by a factor greater than 10MW.

Output representation	%Performance	%Error	% Fatal (>10MW) Predictions
4-bit encoded	98.35	1.65	4.70
1-of-16	98.04	1.96	14.32

Table 5.15 Comparison of the best performance achieved by 4 and 16 output neurons respectively.

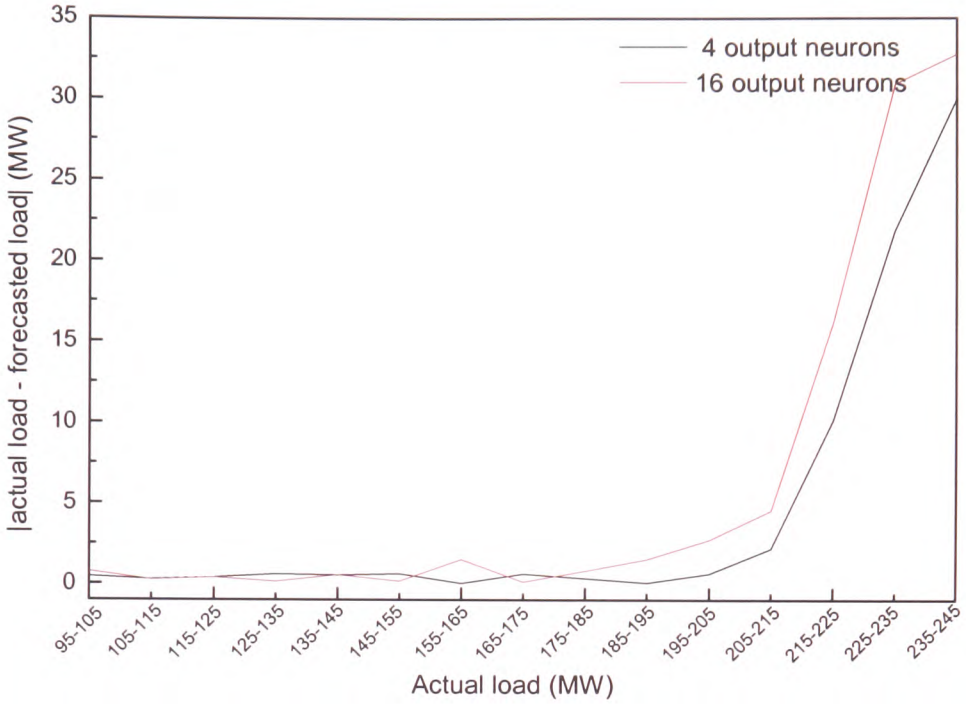


Figure 5.5 Average difference of the predicted load value from the actual as a function of the load range for both 4 and 16 output neurons.

5.8. Results' conclusions-leads to further investigation

- The majority of the faulty predictions are in load values greater than 200MW. Figure 5.6 shows the actual and predicted load values sorted in ascending actual value, while Figure 5.7 illustrates a detailed view of the high load values presented in Figure 5.6.

- Although the majority of the faulty load forecasts are occurring during the winter months, as shown in Figure 5.8, the load demand values which exhibit the larger differences from the actual, by a factor of 20MW or greater, are occurring during the summer (Figure 5.9). This can be easily explained if it is taken into consideration the previously mentioned conclusion *i.e.* the larger

differences from the actual values are occurring during the summer and the fact that the load demand during the summer reaches its annual maximums.

- 30% of the faulty predictions are measured on Fridays as shown in Figure 5.10. This is due to the fact that Friday is the day of the week with the highest load values. On the other hand, the respective 30% of the faulty predictions, which exhibit the larger differences from the actual, by a factor of 20MW or greater, are occurring on Saturdays as illustrated in Figure 5.11. This is mainly a result of the use of the Friday's load values as input to the neural network in order to predict Saturday's demand. Friday, as with all the weekdays, exhibits a totally different load curve compared with weekend load curve, thus the network is unable to produce an efficient prediction.

- Differences in the percentages of the faulty predictions among the months of the year (Figures 5.8 and 5.9) or the days of the week (Figures 5.10 and 5.11) can be attributed to various external factors that influence the load demand of the month or day exactly before the one under consideration. For example January presents a larger percentage of errors than December, a fact that can be possibly attributed to the holiday season in January. Similarly the percentage of faulty predictions larger than 20 MW that is seen on Saturdays can be attributed to the fact that weekends exhibit different behaviour than weekdays.

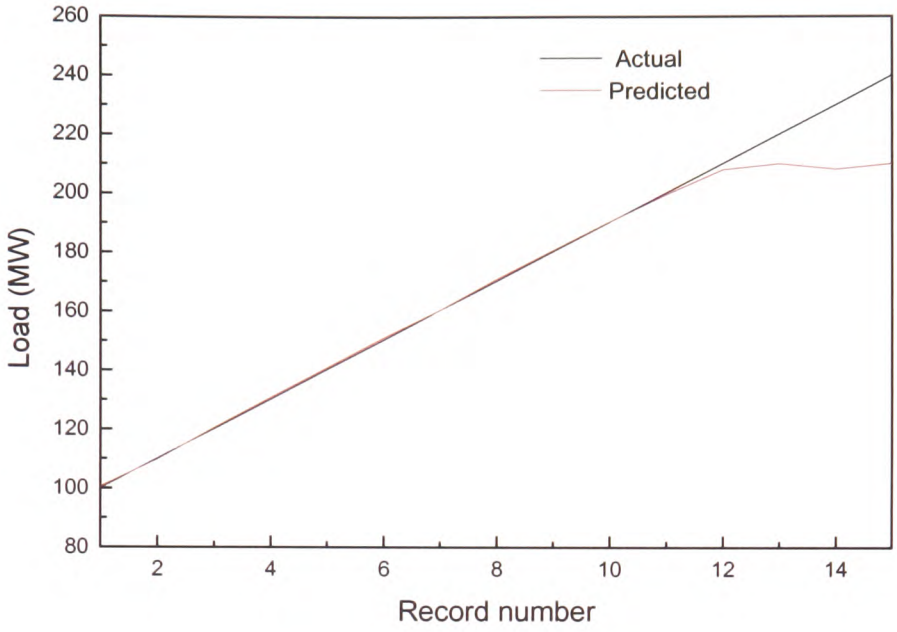


Figure 5.6 Actual and predicted load values sorted in ascending actual load values

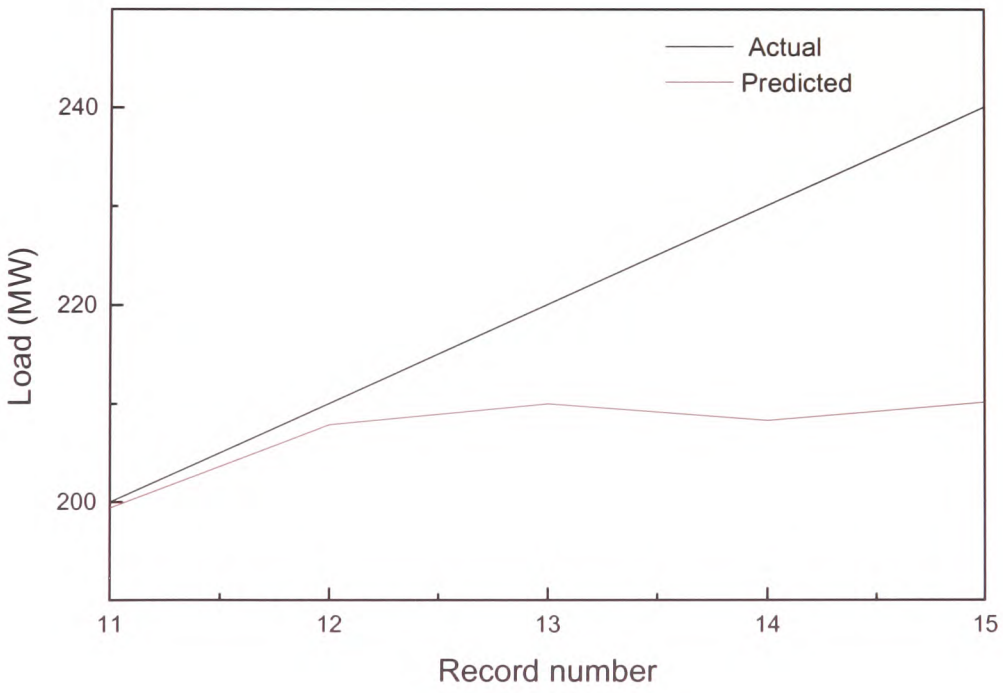


Figure 5.7 A detailed view of the high load values presented in Figure 5.6

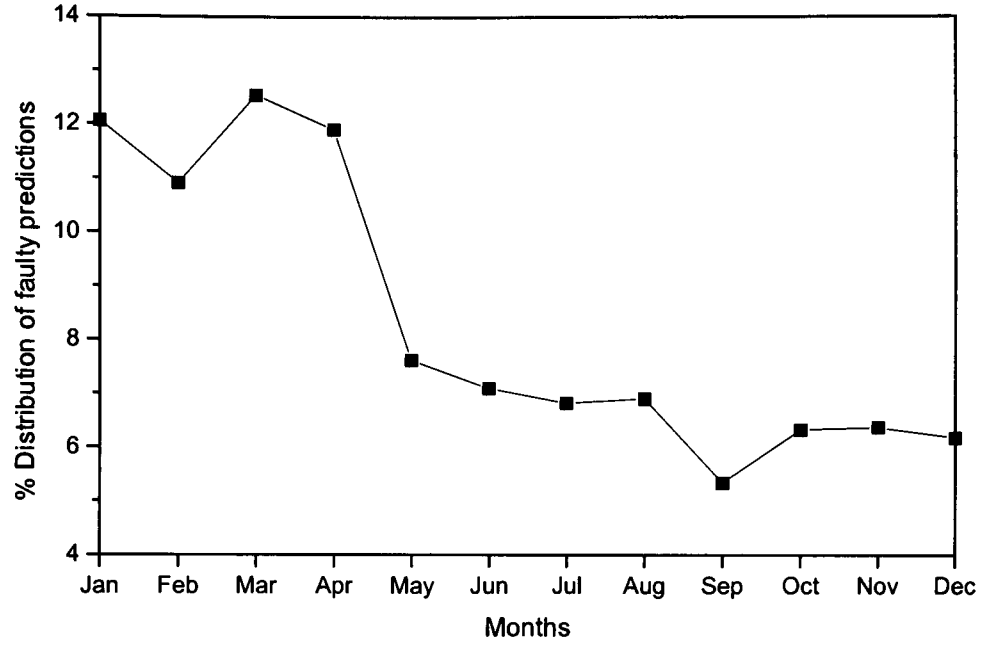


Figure 5.8 % Distribution of the faulty predictions per month

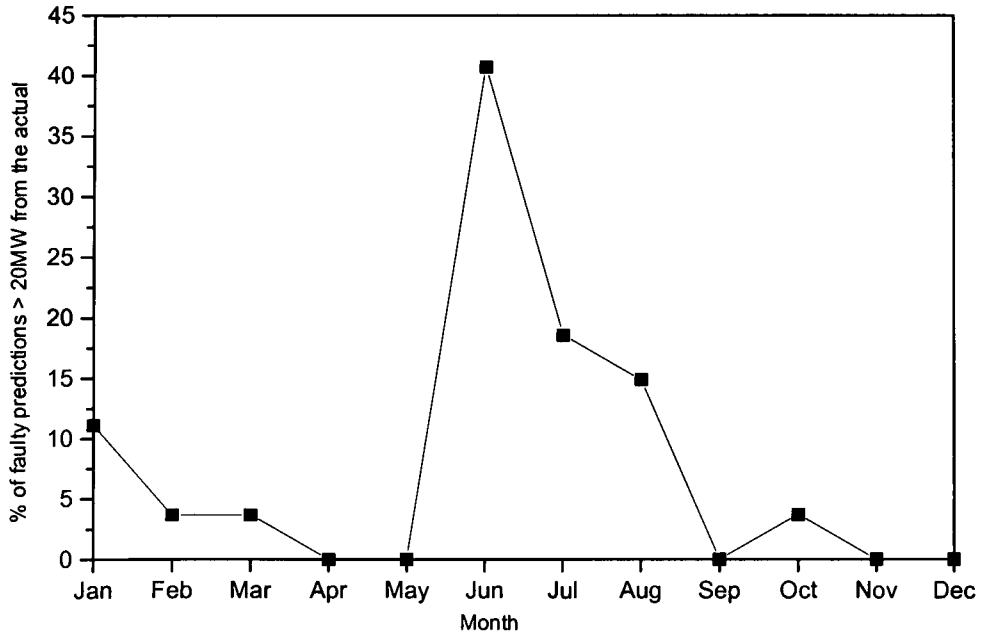


Figure 5.9 Percentage of the faulty load demand values which exhibit the larger differences from the actual, by a factor of 20MW or greater as a function of the month

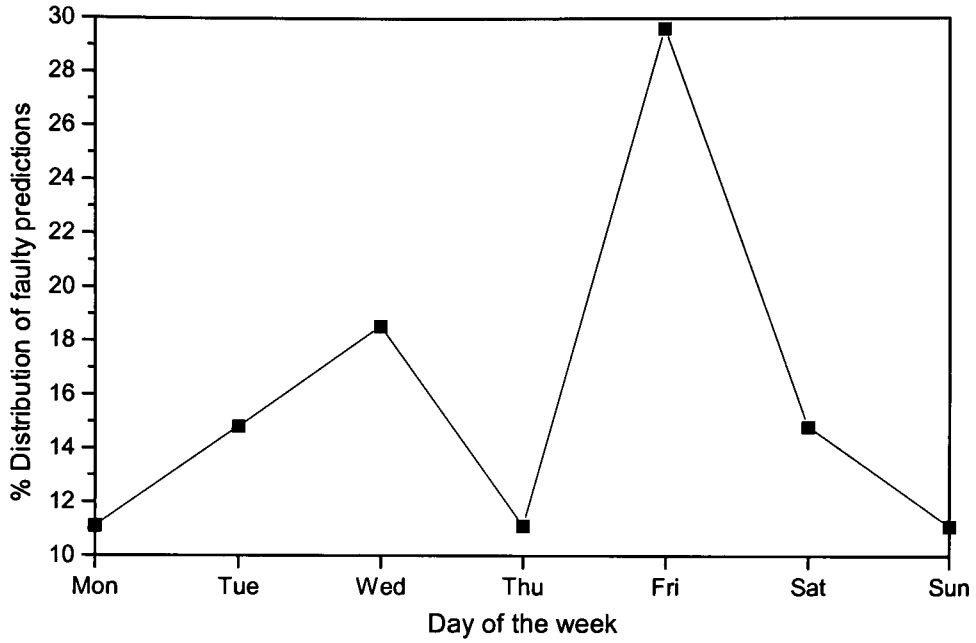


Figure 5.10 % Distribution of the faulty predictions per day of the week

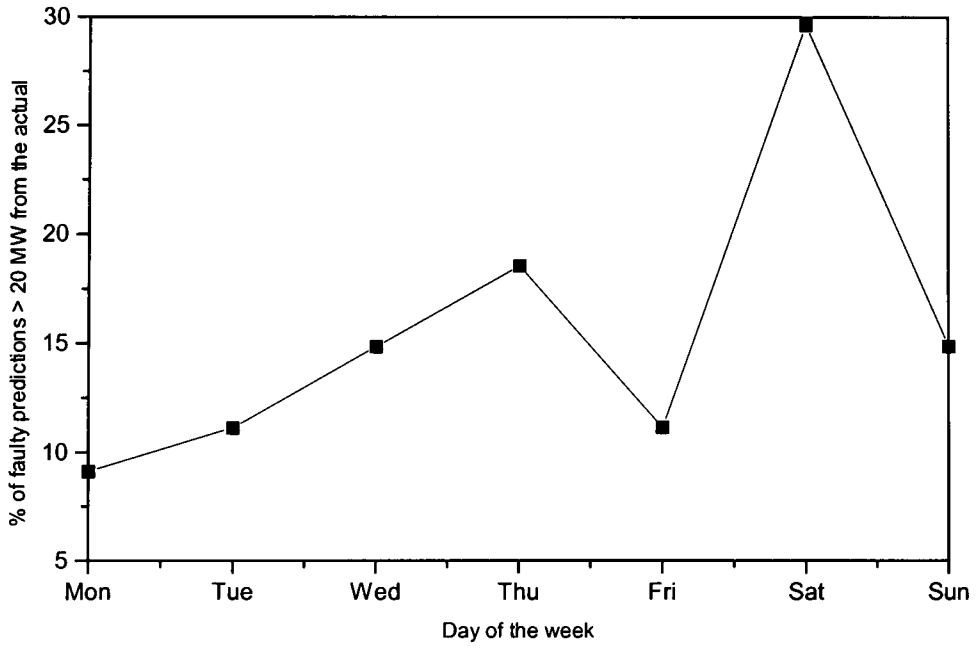


Figure 5.11 Percentage of the faulty load demand values which exhibit the larger differences from the actual, by a factor of 20MW or greater as a function of the day of the week

5.9. Seasonal-separated neural networks

Since the majority of the faulty load demand values which exhibit the larger differences from the actual, by a factor of 20MW or greater are occurring during the summer months (Figure 5.9), another approach is introduced where the summer months' load predictions are achieved with the use of a separate neural network, while another neural network is used for the prediction of the load demand during the period between October and April. To facilitate this, two new training databases are used based on the standard database HTDB1 which contain training patterns from May to September and October to April respectively.

5.9.1. A neural network for an hour ahead load prediction during the summer

The network topology used for the prediction of the summer months' load demand is the one described in Figure 5.1 which is the one with which the best overall performance was achieved up till now. The training database contains 1896 patterns while the database used to evaluate the performance of the network contains 2426 patterns. It can be concluded from Table 5.16 the performance of the network used to forecast the load demand values exclusively during the summer has slightly decreased compared to the performance of the network used to predict the load demand for the whole year.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	486	0.14	2.26	97.74

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	94.35	5.17	0.16	0.32	0

Table 5.16 Neural network results for load predictions during summer from the actual values.

5.9.2. A neural network for an hour ahead load prediction during autumn, winter and spring

The same network was used this time to predict the load demand from October till April. The training database contains 5448 patterns while the database used to evaluate the performance of the network contains 8256 patterns. In this case also the results achieved were at no time better than the ones achieved with the network that was used to forecast the load demand of the whole year. Table 5.17 shows that the faulty measurements have a difference of more than 20 MW from the actual values, which indicates a clear increase compared to the ones achieved by the network used to forecast the load demand for the whole year.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	172	0.18	1.69	98.31

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	93.71	5.38	0.78	0.09	0.04

Table 5.17 Neural network results for load predictions during autumn, winter and spring

The basic conclusion that can be derived from the comparison of the above results with the ones that came from the use of the same neural network for all seasons, is that the faulty predictions are produced by the inability of the neural network to predict the high load values efficiently and it is not a result of the different behaviour of the load curve during the summer. Namely, even when the neural network is trained specifically for the summer load demand patterns the results are the same at best with the ones that are given by the neural network when it is trained using the load patterns of the whole year.

5.10. Daily separated neural networks

As can be observed from Figures 5.10 and 5.11 there is a non-linear distribution of the faulty measurements along the days of the week. The majority of the faulty measurements occurs on Friday, which is a day when load demand maxima are observed. The majority of the faulty measurements that differ more than 20 MW from the actual values occurs on Saturday. So a different approach is presented beneath which uses seven different neural networks, one for each day of the week. From the standard training database (HTDB1) seven new training databases were created, one for each day of the week, and the neural networks were trained by training patterns different for each day of the week. Seven more test databases were created based on the standard test database (HUDB1) so that the networks' performance can be measured. Figure 5.12 presents the architecture of the system while Table 5.18 shows both the training and the test sets that were used to train and evaluate the efficiency of the seven different neural networks.

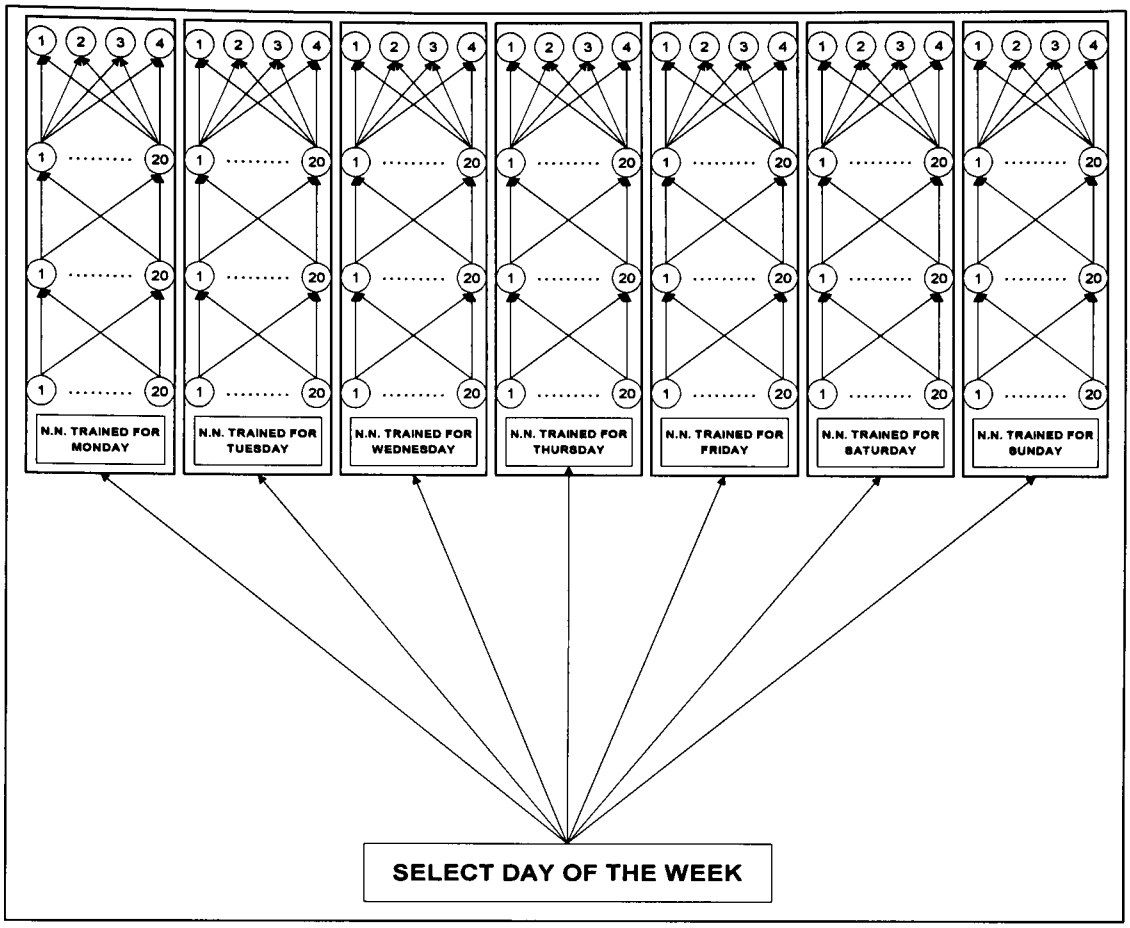


Figure 5.12 System architecture for the implementation of 7 different neural networks, one for each day of the week.

Day of the week	Training patterns	Test patterns
Monday	1310	1704
Tuesday	1224	1704
Wednesday	1228	1820
Thursday	1224	1686
Friday	1246	1692
Saturday	1196	1728
Sunday	1248	1796

Table 5.18 Training and test sets used for the implementation of the 7 day-of-the-week segregated neural networks

5.10.1.Implementation of 7 different neural networks, one for each day of the week

All seven neural networks were trained with the same learning rate and momentum parameters. The results for all seven networks are presented in Table 5.19 while the percentage of the faulty forecasted load values, depending on their difference from the actual, is illustrated in Table 5.20.

Day of the week	Learning rate η	Momentum α	Epochs	MSE	%Error	%Performance
Monday	0.3	0.9	357	0.175	2.17	97.83
Tuesday	0.3	0.9	366	0.160	2.13	97.87
Wednesday	0.3	0.9	346	0.186	1.80	98.20
Thursday	0.3	0.9	368	0.174	2.01	97.99
Friday	0.3	0.9	324	0.210	2.14	97.86
Saturday	0.3	0.9	350	0.198	2.25	97.75
Sunday	0.3	0.9	345	0.184	1.99	98.01

Table 5.19 Neural network results for the 7 day-of-the-week segregated neural networks

Actual – forecasted Load (MW)	Percentage of the faulty forecasted load, depending on their difference from the actual						
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
10	94.97	92.54	95.03	96.67	93.91	94.32	95.24
20	4.10	6.34	3.67	3.13	5.18	5.05	2.94
30	0.93	1.12	1.08	0	0.58	0.63	1.82
40	0	0	0.22	0.20	0.19	0	0
50	0	0	0	0	0.14	0	0

Table 5.20 Percentage of the faulty predicted load values as a function of their difference from the actual for the day-of-the-week segregated neural networks

Results presented so far strongly suggest that there is no actual improvement to the system's performance compared to that of the network that was used to forecast the load demand for every day of the week.

5.11. Further improvement of the results

The conclusion from the above presented network architectures, and corresponding results, is that the standard network presented (Figure 5.1) provides extremely reliable results in every case apart from when high load demand values are observed. This observation indicates that there is a lack of an adequate number of training patterns for the corresponding intervals. This observation is proved correct by Figure 5.13, which illustrates the distribution of training patterns according to load range. The lack of training patterns for high load ranges is attributed to the fact that only for a few days during a year does the demand reaches such high values. The problem was overcome only when the Public Power Corporation provided new data-sets for the years 1996 (second half), 1997 and 1998 (first half). These data were only available late in the course of this research.

The same preprocessing methods as described in Chapter 4 were applied to these data. Then the database HTDB1 was expanded by the addition of training patterns which represent load values greater than 200 MW. Thus a training database was created with 8885 training patterns. The pattern distribution according to load range of this improved database is shown in Figure 5.14.

The neural network described in Figure 5.1 was used again since it was the network that achieved the best overall performance and since it also provides a common base to compare the results presented here with the ones achieved with the use of the standard database. For the same reason the values of 0.3 and 0.9 were kept for the learning rate and momentum parameters respectively. Table 5.21 shows that better results are achieved, compared to the ones shown in Table 5.6, with the use of the improved training database not only regarding the average

error but also the fact that the faulty measurements categorised according to their difference from the actual values are highly concentrated in the [0,10] MW interval. This means that the vast majority of the faulty measurements is really close to the actual values.

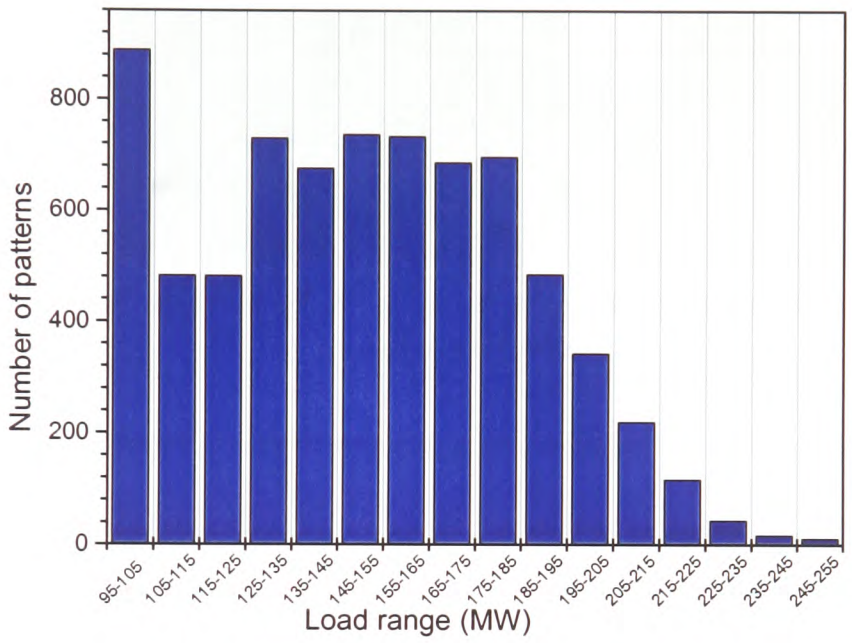


Figure 5.13 Distribution of patterns according to load range (Standard training database)

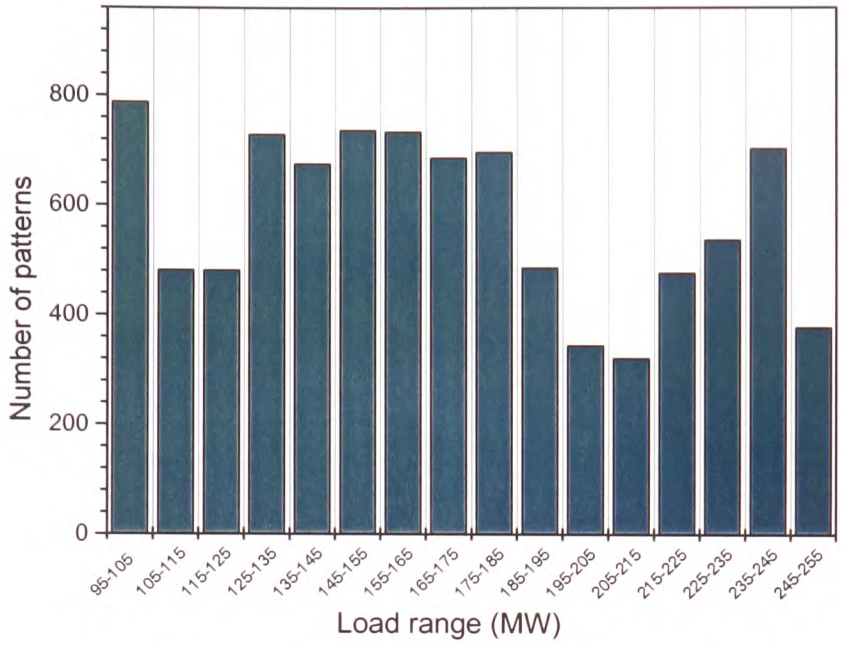


Figure 5.14 Distribution of patterns according to load range (improved training database)

Figure 5.15 illustrates the load prediction improvement achieved with the improved training database, especially in the high load values, over the standard training database. Figure 5.16 illustrates the actual and predicted load values using both the standard and improved training databases, sorted in ascending actual load value. As can be easily noted, there is a substantial improvement in the prediction when the newly created database is used compared to the prediction when the standard training database (HTDB1) is used.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	79	0.15	1.47	98.53

| actual load - forecasted load |	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	97.06	2.58	0.31	0.04	0.01

Table 5.21 Neural network results for load predictions using the improved training database.

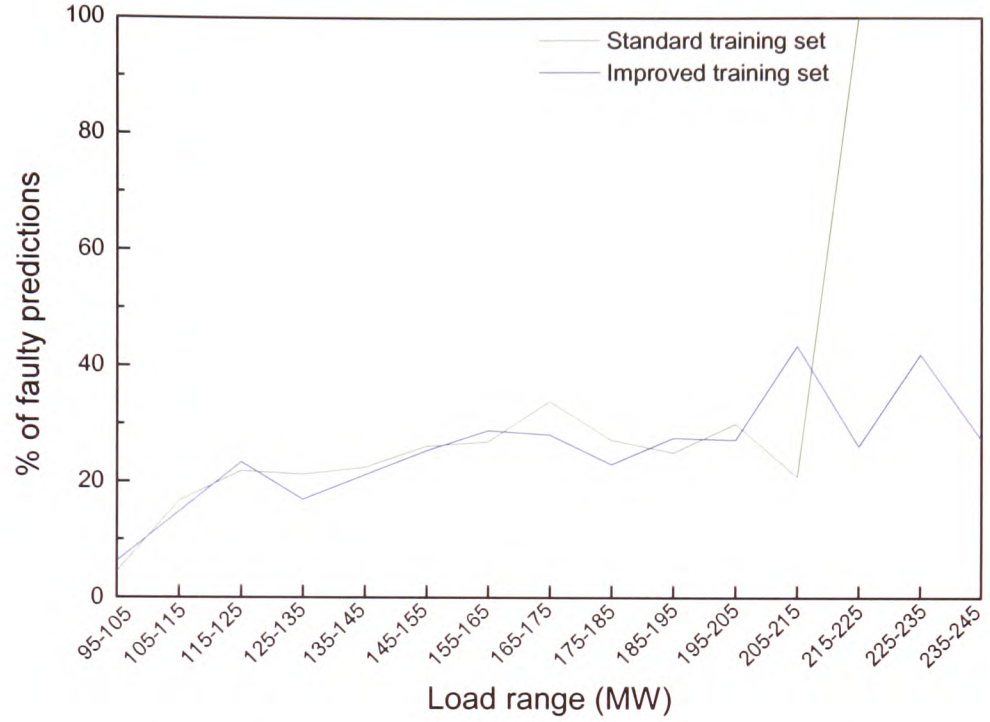


Figure 5.15 Test patterns' % of faulty predictions according to load range

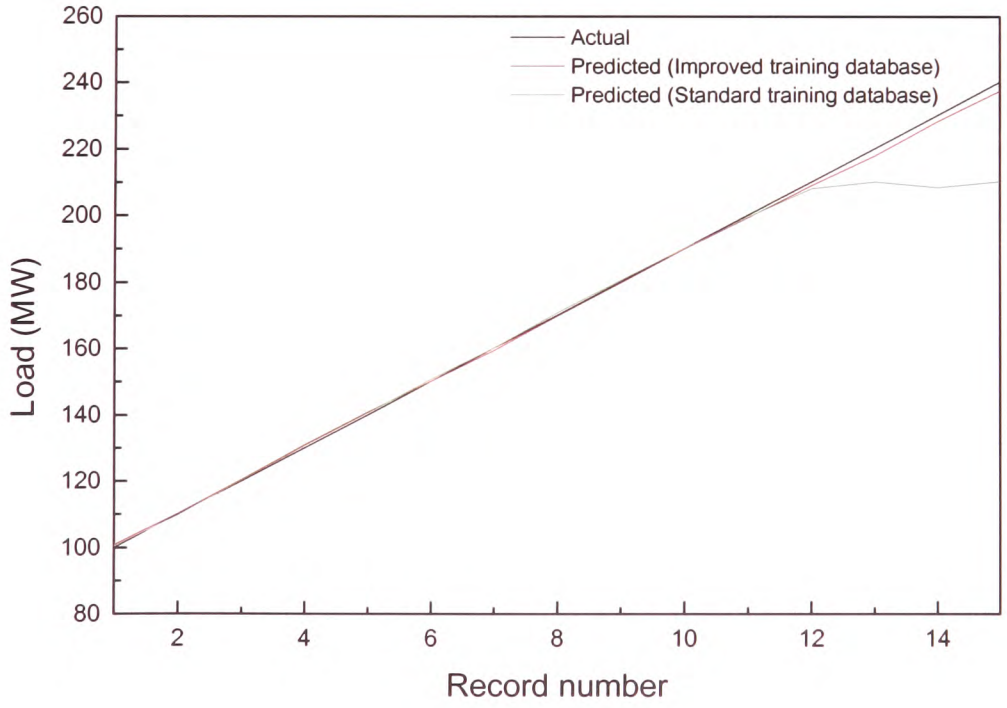


Figure 5.16 Actual and predicted load values sorted in ascending actual load values.

5.12. A 24-output neural network for a whole day load forecast

For an electric power production unit, an equally important prediction to the one hour ahead load demand is the prediction of the next day hourly load forecast. This gives a first impression of the load demand for the next day and helps substantially the correct daily programming of the production units.

A 61 input neurons network is selected for the prediction of the one day ahead hourly load demand. A detailed description of the input vector is presented in Table 5.22. Two hidden layers were used with 50 neurons each. The output layer has 24 neurons, one for each hour of the day. The network architecture used is presented in a graphical form in Figure 5.17.

NODE	DESCRIPTION
1-24	The hourly load demand a day before the prediction day.
25-48	The hourly load demand a week before the prediction day.
49-50	Minimum and maximum temperature values one day before the prediction day
51-52	Minimum and maximum temperature values one week before the prediction day
53-54	Minimum and maximum forecast temperature values at the prediction day
55-61	One-of-N representation of the day of the week

Table 5.22 hourly load prediction for one day ahead. Summary of Input variables

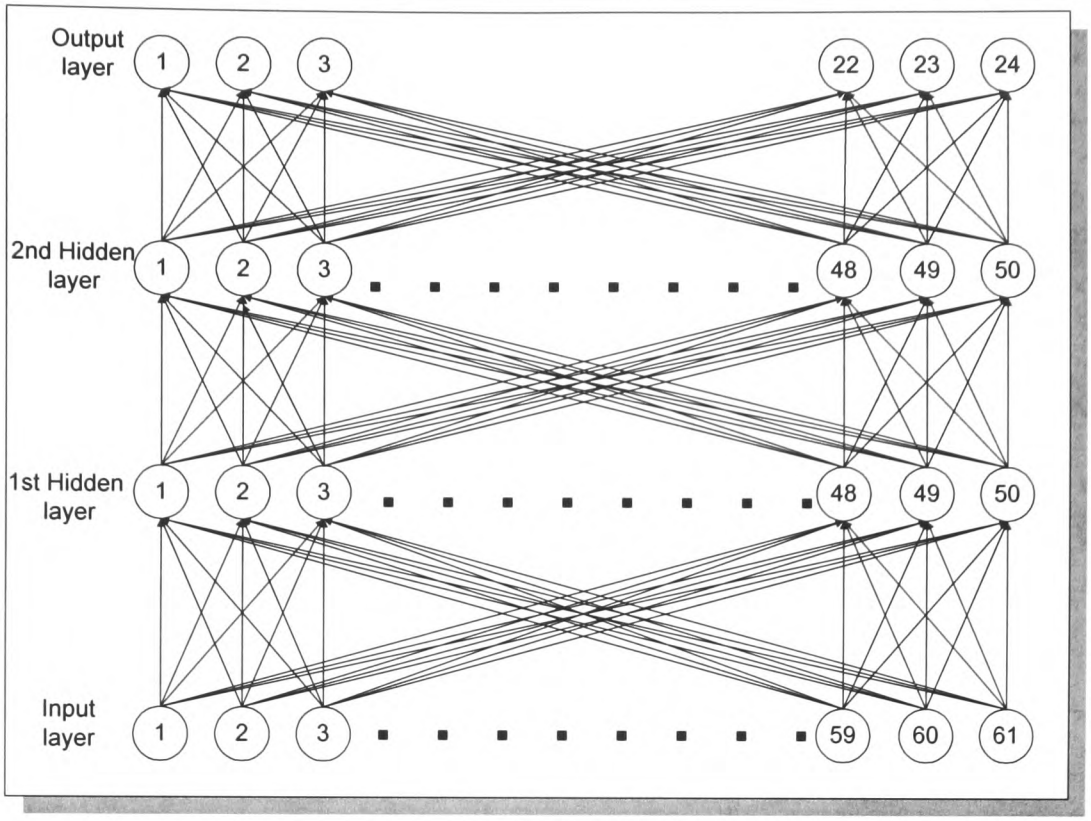


Figure 5.17 architectural graph of the neural network used for 24-hour ahead hourly load prediction

The neural network was trained for the one day ahead hourly load forecasting with the training database DTDB1 which contains 320 patterns derived from the year 1994 data. The values that produced the best result at the one hour ahead load forecasting were used for the learning rate η and momentum α parameters, *i.e.* 0.3 and 0.9 respectively (Table 5.23). Further experiments were carried out though so that the effect of these parameters to the overall network efficiency could be determined, the case of η=0.5 and α=0.8 is presented in Table 5.24. Finally, the test database DUDB, which contains 534 patterns created from 01/01/1995 to 30/06/96, was used to evaluate the performance of the network.

Tables 5.23 and 5.24 show that increasing the learning rate parameter has no substantial effect on the performance, the percentage of the faulty measurements that differ 20 MW or more from the actual load demand value however increases notably.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	78	0.049	3.66	96.34

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	83.60	13.38	2.91	0.11	0

Table 5.23 Neural network results for the hourly load prediction for one day ahead, learning rate equals 0.3, momentum parameter equals 0.9.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.5	0.8	62	0.051	3.69	96.31

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	76.83	18.61	4.25	0.20	0.11

Table 5.24 Neural network results for the hourly load prediction of a whole day ahead, learning rate equals 0.5, momentum parameter equals 0.8.

It is also interesting to perform experiments based on neural networks with reduced complexity. A second network topology with 30 hidden neurons was used. With this topology the network performance slightly degraded compared to the one achieved with the corresponding 50 hidden neuron network (Table 5.25). Furthermore, the percentage of faulty measurements that differ by 20 MW or more from the actual increases substantially in this case.

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	96	0.048	3.90	96.10

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	74.69	20.09	4.10	0.62	0.5

Table 5.25 Neural network results for the hourly load prediction of a whole day ahead - 30 neurons on each hidden layer.

Following the same procedure as in paragraph 5.11 the training database was improved by the addition of training patterns from years 1997 and 1998. The architecture and the training parameters were kept the same. The improved training database contained 446 patterns and for testing the same database was used as before. The results in this case are again better than the ones achieved with the standard database for the one day hourly load forecast (Table 5.26).

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	112	0.042	2.60	97.4

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	82.8	11.2	3.4	2.3	0.3

Table 5.26 Neural network results for the hourly load prediction of a whole day ahead - training with the improved database.

5.13. Noon and evening load peaks forecasting

Another important factor for the production and distribution departments of a power system is the daily peak load values. The valid prediction of the daily maxima helps the correct programming and distribution of the load which will be produced by the production units. It also helps programming the maintenance of the production units which has to take place during days that the forecasted demand would be covered by the remaining units.

In the island of Crete, two load peaks are observed in the daily load curve. The first load peak is observed between 12:00 and 14:00 depending on the season, while the second is between 19:00 and 21:00 depending on the season. A typical daily load curve is presented in Figure 5.4.

Two different neural networks were used to predict the noon and the evening peak respectively. The network topology of both neural networks is presented in

Figure 5.18. both networks consist of two hidden layers with 10 neurons in each layer. The input layer contains 10 neurons, while the output layer is represented with 4-Gray encoded neurons which correspond to the 16 load ranges used in both the hourly load demand prediction and the one day ahead hourly load demand prediction.

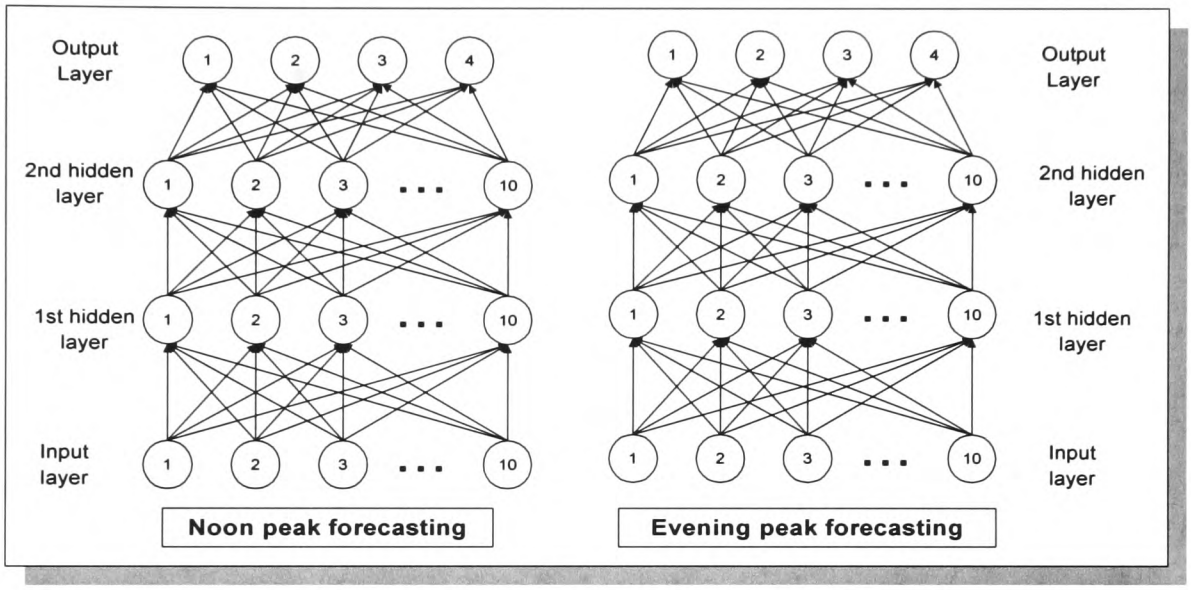


Figure 5.18 Architectural graph of the neural networks used for noon and evening peak load forecasting.

5.13.1. Noon peak prediction

For the noon peak prediction, the NPTDB1 training database was used which contains 159 training patterns, while for testing the NPUDB test database was used which contains 448 patterns. The input vector used for the noon peak prediction is illustrated in Table 5.27.

NODE	DESCRIPTION
1	Noon peak load value a day before the prediction
2	Noon peak load value a week before the prediction
3-4	Minimum and maximum temperature values one day before the prediction day
5-6	Minimum and maximum temperature values one week before the prediction day
7-8	Minimum and maximum forecast temperature values at the prediction day
9	The day of the week
10	The month of the prediction

Table 5.27 Summary of input variables used for noon peak load forecasting

A learning rate of 0.3 and a momentum parameter of 0.9 were used for the training. The forecast performance reached a very encouraging value (Table 5.28).

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	228	0.40	1.78	98.22

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	97.26	2.34	0.3	0.1	0

Table 5.28 Neural network results for the noon peak load prediction.

5.13.2.Evening peak prediction

The structure of the neural network used for the evening load forecasting is the same as that used for the noon peak load forecasting. The EPTDB1 training database that used for the training consists of 159 patterns while the EPUDB

database which used for the network validation consists of 448 patterns. The input vector used for the evening peak prediction is shown in Table 5.29.

NODE	DESCRIPTION
1	Evening peak load value a day before the prediction
2	Evening peak load value a week before the prediction
3-4	Minimum and maximum temperature values one day before the prediction day
5-6	Minimum and maximum temperature values one week before the prediction day
7-8	Minimum and maximum forecast temperature values at the prediction day
9	The day of the week
10	The month of the prediction

Table 5.29 Summary of input variables used for evening peak load forecasting

The same learning parameters were used for the evening peak load prediction as for the noon peak. The network performance is again satisfactory, especially when the percentage of the faulty measurements that differ by 20 MW or more from the actual value is considered. In that case almost every faulty measurement differs only by around 10 MW from the actual value (Table 5.30).

Learning rate η	Momentum α	Number of epochs	Mean square error (MSE)	%Error	%Correct
0.3	0.9	176	0.50	1.72	98.28

|actual load - forecasted load|	10 MW	20 MW	30 MW	40 MW	50 MW
% of the faulty predicted loads, depending on their difference from the actual	98.24	1.56	0.1	0.1	0

Table 5.30 Neural network results for the evening peak load prediction.

6. CONCLUSIONS AND FUTURE WORK

6.1. Synopsis of the research

This thesis describes the research and analysis carried out in order to implement a system for short term load forecasting for the isolated power system of the island of Crete.

The system is largely divided into two basic parts:

- Data preprocessing including:
  - The introduction and implementation of a special data preprocessing methodology, used to reject faulty measurements and normalise the load demand values.
  - The introduction and testing of a new neural network input and output representation, based on the nonlinear representation of the nominal values and Gray code respectively.
  - The performance of extensive studies for the importance of various factors such as temperature, season, day of the week, *etc.* to the load demand.
- Construction of the neural networks, where various network topologies based on multilayer perceptrons with error back-propagation learning rule are used in order to identify the architecture that gives the best result. Extensive studies were also performed to see the effect of factors such as learning rate, momentum, number of the training patterns, *etc.*

The results achieved are very promising, although the Power System of Crete has unique characteristics that impede load forecasting. The results are rated among the top ones found in the bibliography, although a direct comparison is not always proper due to the diversity of the nature of the different power systems (see chapter 2). Table 6.1 summarises the resulting average forecast error achieved for hourly, daily and peak load forecasting. An element not shown in Table 6.1 is

that the percentage of the errors lying in the next output range (higher or lower) reaches 97.06%. This means that the vast majority of the errors are extremely close to the actual load value so that even them can be used for forecasting.

Forecast type	%Average error
One hour ahead	1.47
One day ahead	2.60
Noon load peak	1.78
Evening load peak	1.72

Table 6.1 Summary of the load forecast errors

6.2. Concluding remarks

The performance of the system depends on various different parameters. Normalising and encoding the input and output patterns prior to their application to the neural network can actually provide all the parameters related to the load demand in an efficient way for use by the neural network.

Extensive studies were performed on the effects of the learning rate and the momentum parameter. The performance of the network in the case of a learning rate equal to 0.3 and a momentum parameter equals 0.9 reaches the highest value. The network efficiency is affected negatively when using a learning rate greater than 0.5. The momentum parameter should in a value greater than 0.8 in order to achieve an efficient network performance. An adaptive learning algorithm was also introduced but had minimal effects on the system performance.

The network performance was very slightly affected by the dimension and the number of the network layers, while the exact same factors had a major effect on the training time. Moreover a more noticeable effect on the training time was produced by the increase of the training patterns and the reduction of the learning rate parameter.

The number of the training patterns has, in general, minimal effect on the performance. Only in the case of a significant reduction in the number of the training patterns was there a substantial increase in the forecast error.

The system performance depends mainly on the quality and the number of the training patterns for each load range. The selection of the training cases significantly affects the forecasting of results. The standard training database (HTDB1) with one year period patterns did not contain enough patterns for the high load values (>200 MW). That problem was solved with the construction of an improved training database created by extending the standard database with training patterns derived from data sets from different years. That way the minimum forecast error was achieved.

Load demand is strongly influenced by the seasonal factors. An abnormal increment of the load demand is noted as a result of the increased population due to tourism during the summer months. Thus the load curve during the summer months exhibits a totally different behaviour compared with the one during the rest of the year. Consequently, a different approach was also examined with two different neural networks, one for the summer months and the other for the rest of the year but the results were not satisfactory.

Another important factor related to the load demand is the day of the week. The load curve during the weekends is significantly lower than the one during the weekdays. In order to overcome the incompatibility between the weekdays and the weekends seven different neural networks, one for each day of the week, were implemented but neither in this case were the results satisfactory.

In both cases the results were not satisfactory due to the small and definitely not representative number of training patterns for the load ranges greater than 200MW, which contributed the most to the resulting error.

6.3. Suggestions for future work

Each power system has its own characteristics. Since the load profile is dynamic in nature with temporal, seasonal and annual variations, many structures for inputs and neurons need to be tested experimentally to get the best forecasting result.

There are a number of recommendations on different approaches for further investigation:

1. An extended training database that consists of representative data for every load range will in most cases produce even better results. This means that the availability of adequate historical data sets for training and evaluation of performance is a crucial factor.

2. The development of separate neural networks based on the various parameters related to the load demand. Hence each neural network can be considered as a better representation of particular scenarios which allow the mapping of the historical features of load in a more representative way. An expert system could be used for the categorisation of the front end so as to determine the best and most appropriate neural network for the final prediction. This might be achieved by an expert system capable of decomposing the problem through a number of hierarchies using formal logical steps.

3. Unsupervised learning Kohonen's self-organising future maps can be used for classifying those days which have similar hourly load patterns and belong to the same day pattern, whereas the prediction itself will be made with the use of multilayer perceptron networks each trained especially for the generalisation of the specific day pattern.

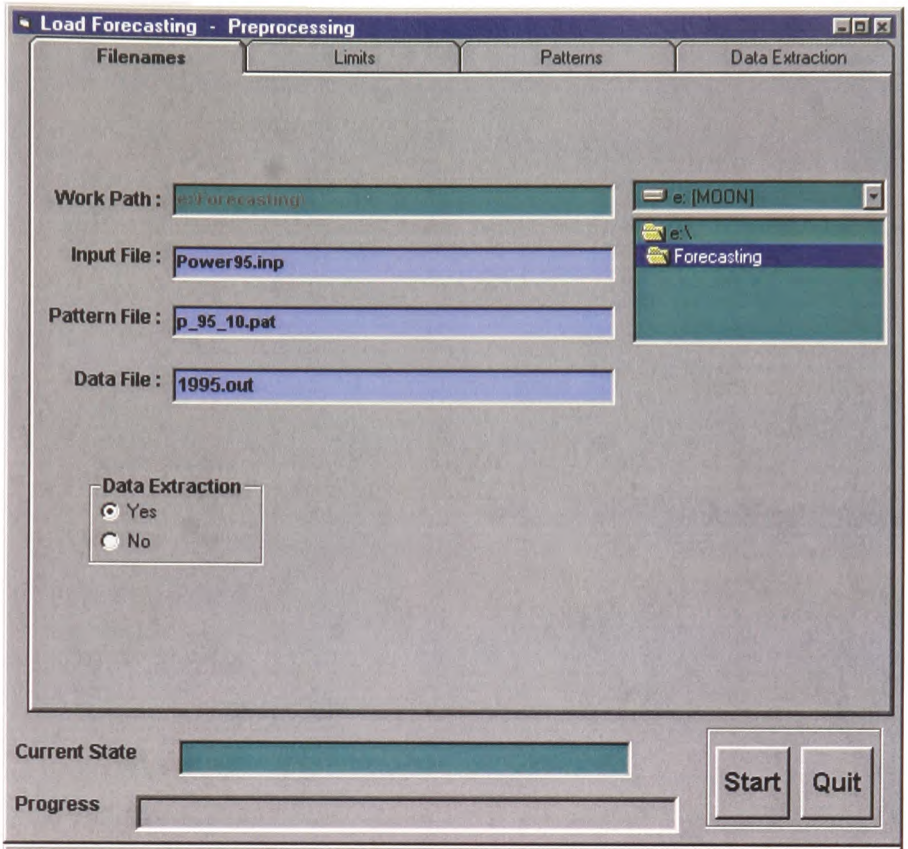
4. The results also show that the predicted load is always lower than the actual value in the higher load values (> 200 MW) while exactly the opposite happens in the lower load values (around 100 MW) where the predicted load usually exceeds the actual. A different load range separation that will take into account this property of the system could possibly lead to better results.

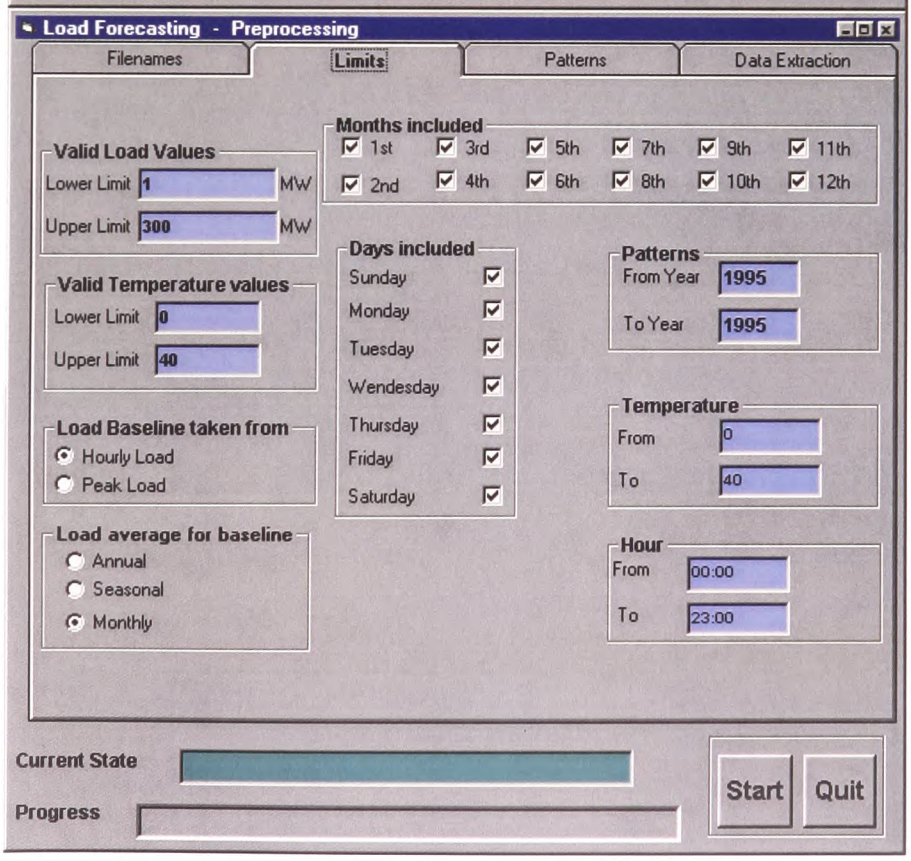
Appendix A

A.1 The preprocessing software

The implementation of the data preprocessing described in Chapter 4 was achieved by the use of a software tool developed especially for the load data preprocessing. It provides a rich set of operations for data pre and postprocessing. Various data types are supported, including symbols, numbers, vectors of numbers, one-of-N codes, and binary codes. Various representation filters are developed so that numeric values can be converted into the coded types for presentation to the neural network and can be converted back from codes to numbers on the output side. All the pre and postprocessing parameters can be changed by the user in a user-friendly graphical environment. Figure A.1, A.2, A.3 and A.4 illustrate some of the forms of the program.

The source data must be in a plain ASCII file as taken from the local Power Corporation. The output files can be in various ASCII file formats, including comma separated or SNNS [47] neural network simulator pattern file format, which is the network simulator mostly used for the implementation of the neural networks in this work.


Load Forecasting - Preprocessing
Filenames
Limits
Patterns
Data Extraction
Work Path :
e: [MOON]
e:\
Forecasting
Input File :
Power95.inp
Pattern File :
p_95_10.pat
Data File :
1995.out
Data Extraction
Yes
No
Current State
Progress
Start
Quit


Load Forecasting - Preprocessing
Filenames
Limits
Patterns
Data Extraction
Valid Load Values
Lower Limit 1 MW
Upper Limit 300 MW
Months included
1st 3rd 5th 7th 9th 11th
2nd 4th 6th 8th 10th 12th
Valid Temperature values
Lower Limit 0
Upper Limit 40
Days included
Sunday
Monday
Tuesday
Wendesday
Thursday
Friday
Saturday
Patterns
From Year 1995
To Year 1995
Load Baseline taken from
Hourly Load
Peak Load
Temperature
From 0
To 40
Load average for baseline
Annual
Seasonal
Monthly
Hour
From 00:00
To 23:00
Current State
Progress
Start
Quit

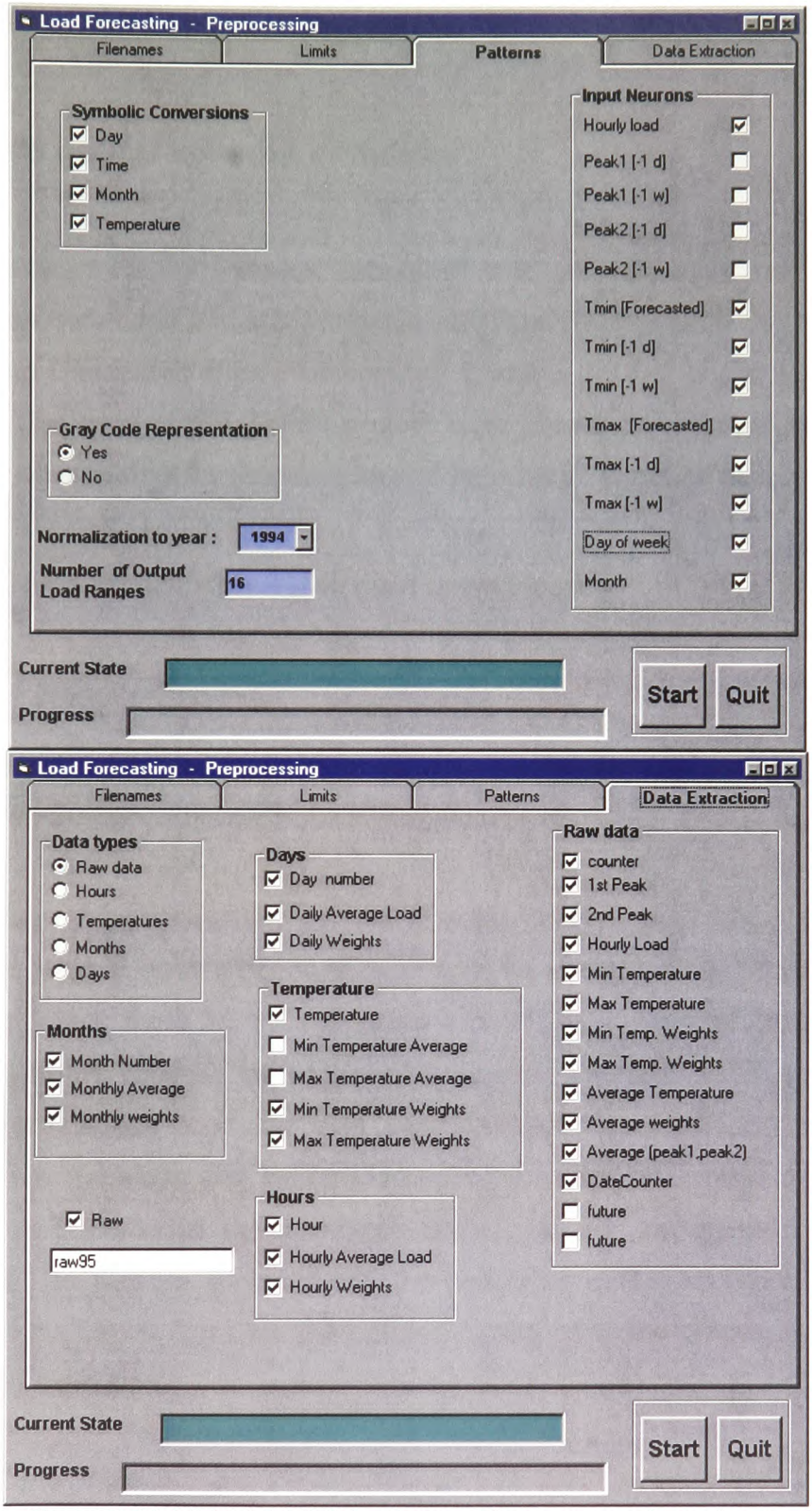


Figure A.1 (a),(b),(c) & (d) Setup forms of the preprocessing software.

Appendix B

B.1 SNNS neural network simulator.

SNNS (Stuttgart Neural Network Simulator) is a software simulator for neural networks on both Unix and MS Windows platforms developed at the Institute for Parallel and Distributed High Performance Systems (IPVR) at the University of Stuttgart. The goal of the SNNS project is to create an efficient and flexible simulation environment for research on and application of neural nets.

The SNNS simulator consists of two main components:

1) simulator kernel written in C

2) graphical user interface under X11R4 or X11R5

The simulator kernel operates on the internal network data structures of the neural nets and performs all operations of learning and recall. It can also be used without the other parts as a C program embedded in custom applications. It supports arbitrary network topologies and supports the concept of sites. SNNS can be extended by the user with user defined activation functions, output functions, site functions and learning procedures, which are written as simple C programs and linked to the simulator kernel. All the training and validation procedures can also be achieved with the use of an additional batch interpreter interface to the kernel that allows integrated control of all the processes and easy background execution.

Currently the following network architectures and learning procedures are included:

- Back-propagation (BP) for feedforward networks
- vanilla (online) BP
- BP with momentum term and flat spot elimination
- batch BP
- Counterpropagation
- Quickprop
- Backpercolation 1
- RProp
- Generalized radial basis functions (RBF)
- ART1
- ART2
- ARTMAP
- Cascade Correlation
- Recurrent Cascade Correlation
- Dynamic LVQ
- Back-propagation through time (for recurrent networks)
- Quickprop through time (for recurrent networks)
- Self-organizing maps (Kohonen maps)
- TDNN (time-delay networks) with Back-propagation
- Jordan networks
- Elman networks and extended hierarchical Elman networks
- Associative Memory

The graphical user interface XGUI (X Graphical User Interface), built on top of the kernel, gives a 2D and a 3D graphical representation of the neural networks and controls the kernel during the simulation run. In addition, the 2D user interface has an integrated network editor which can be used to directly create, manipulate and visualise neural nets in various ways.

B.2 Training and test patterns' format

The SNNS data files have a header component and a data component. The header defines how many patterns the file contains as well as the dimensionality of the input and output vector. The files are saved as ASCII text. An example is given if Figure B.1

```
SNNS pattern definition file V3.2
generated at Mon Oct 12 11:37:00 1998


No. of patterns : 7344
No. of input units : 20
No. of output units : 4

#Input 96-01-01  1
 0.554  0.456  0.366  0.263  0.387  0.306  0.230  0.134  0.456  0.367  0.279  0.182
 0.721  0.740  0.713  0.688  0.709  0.669  0.730  0.235
#Output 94-01-10
0 0 0 1
#Input 96-01-01  2
 0.456  0.366  0.263  0.157  0.306  0.230  0.134  0.081  0.367  0.279  0.182  0.120
 0.721  0.740  0.713  0.688  0.709  0.669  0.730  0.121
#Output 94-01-10
0 0 0 0
#Input 96-01-01  3
 0.366  0.263  0.157  0.101  0.230  0.134  0.081  0.064  0.279  0.182  0.120  0.088
 0.721  0.740  0.713  0.688  0.709  0.669  0.730  0.055
#Output 96-01-01  4
0 0 0 0
...
```

Figure B.1 SNNS pattern file diagram

PUBLISHED PAPER

Alexandros Manousakis, George Papadourakis, Andrew Ware, "Electric Load Forecasting with the use of Artificial Neural Networks", *Proceeding: MEDPOWER '98, Nicosia, Cyprus, November 1998.*

Electric Load Forecasting with the use of Artificial Neural Networks

Alexandros Manousakis[1,2&3], George Papadourakis[2], Andrew Ware[1]
Division of Mathematics and Computing, University of Glamorgan, UK[1]
Department of Science, Technological Educational Institute, Heraklion, Crete, Greece[2]
Datacreta S.A., Heraklion, Crete, Greece[3]

Abstract

Load Forecasting plays a key role in the operation and planning of electric power systems. The current research work is aimed at designing a model based on artificial neural networks that is capable of forecasting the hourly load demand for the isolated power system of the island of Crete, Greece. Our interest is focused to the short term load forecasting but middle term load forecasting will also be examined.

Introduction

Electric load forecasting plays a key role in the operation and planning of electric power systems. Long term electric load forecasting is used for economic planning of new generating capacity and transmission networks, while the short term predictions help in the day-to-day operation and scheduling of the power system [1]. In addition, the ability to forecast electric system load can also provide valuable information for possible energy interchange with other utilities. Accurate information for future loads is also useful in detecting many vulnerable situations in advance, if applied to system security assessment problem.

Forecast error in load prediction result in increased operating costs. On the one hand, underprediction of load result is a failure to provide the necessary reserves which translates to higher costs due to the use of expensive peaking units. Overprediction of load, on the other hand, results in an unnecessary increase of reserves and hence operating cost.

The classical load forecasting models mathematically consist, of the following components: a) basic load, b) weather-dependent components, c) noise or residual components and d) special effects such as holidays, major strikes *etc.*

According to this model there is a fully non-linear functional correlation between all the above factors and the load patterns. This relationship has to be represented to achieve the capability for an efficient prediction.

Previous approaches on electric load forecasting fall in general into the following three categories:

1. The load pattern is treated as a time series signal and the future loads are predicted by using series analysis techniques [2-4].
2. A predetermined functional relationship between the load and the weather variables used for the prediction of the future load patterns by inserting the weather information [5-8].
3. The use of artificial neural networks to combine the time series and the regression approach [9-15].

The multilayer artificial neural networks possess a number of properties, which make them an attractive alternative choice. The advantage of artificial neural networks over the statistical models lie in their ability of realization of the complex non-linear relationship between the predicted values and indicators such as historic data (data collected in the past) and exogenous factors. Neural networks are trained to form a non-linear model of the underlying system, which is able to generalize new cases that are not part of the training data.

In this paper a new approach to the load forecasting problem of the isolated power system of island of Crete, Greece based on artificial neural networks is presented. Extensive studies for the importance of various factors such as temperature, season, day of the week to the load demand are also performed. A special data preprocessing methodology is introduced and a new neural network output representation is utilized. A variety of combinations of input patterns are used to examine the effect, in terms of the forecasting error, on the prediction of hourly and daily peak load.

The organization of the paper is as follows: Section 2 presents an overview of established load data analysis and data preprocessing. This is followed by an introduction to artificial neural networks and data preprocessing procedures performed specifically for the neural network optimization. The neural networks architectures used are presented in section 4 along with the results followed by conclusions.

General Data Analysis

The historical load data were obtained from the Public Power Corporation and span two and a half-years, beginning from 1994. Hourly load raw data along with two daily load peaks were available for the above time period. Daily minimum and maximum temperature measurements were also provided. Figure 1 illustrates the overall daily load demand for the years 1994 and 1995. The isolated power system of the island of Crete was selected because of the following special properties that it carries:

During the summer there is an abnormal increment of the load demand as a result of the increased population due to tourism. Therefore, the summer load demand (days 170-270) exhibits totally different behaviour than the rest of the year - as shown in Figure 1.

The excessive increase of the load demands during the summer often exceeds the available load demand, resulting in blackouts as indicated by minimum values in Figure 1.

The average annual load demand increases substantially every year due to the increasing development factor at the island caused by industry and tourism [17].

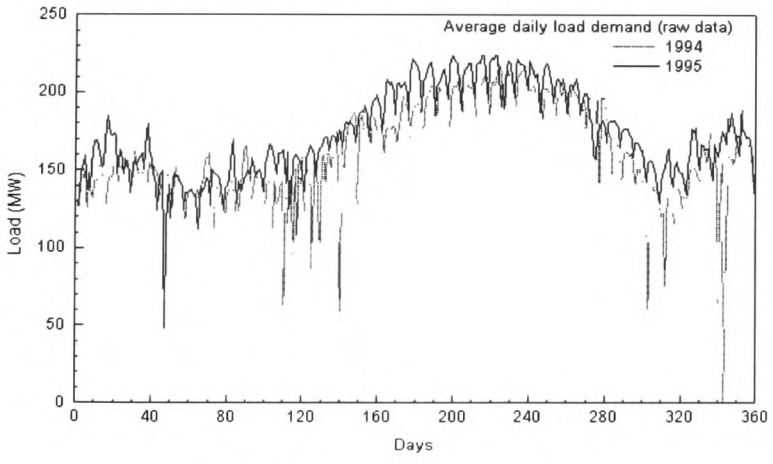


Figure 1: Average Daily Load Demand for the years 1994-1995

In order to overcome the heterogeneous average annual load increment, the monthly average load L_{av} is first calculated for every year including the base year

L_{bav}. Then the raw load data L were normalized to a base load year according to the following equation:

$$L_n = L \cdot \frac{L_{bav}}{L_{av}} \quad (1)$$

where L_n is the normalised daily load to the base year.

In order to improve the data quality, faulty and abnormal measurements were carefully extracted from the data library using rejection algorithms (Chauvenet criterion) [18]. These techniques basically use the previous load data to calculate a typical load curve for everyday of every season. A measurement is characterised as faulty if the percentage of the difference from the typical value is greater than a certain value. These defective measurements are due to blackouts during the summer or due to wrong measurements. Removing these measurements will not harm the generality, however it is believed that by keeping them may lead to faulty prediction. The normalised daily load L_n to the base year after the extraction of faulty and abnormal measurements is shown in Figure 2.

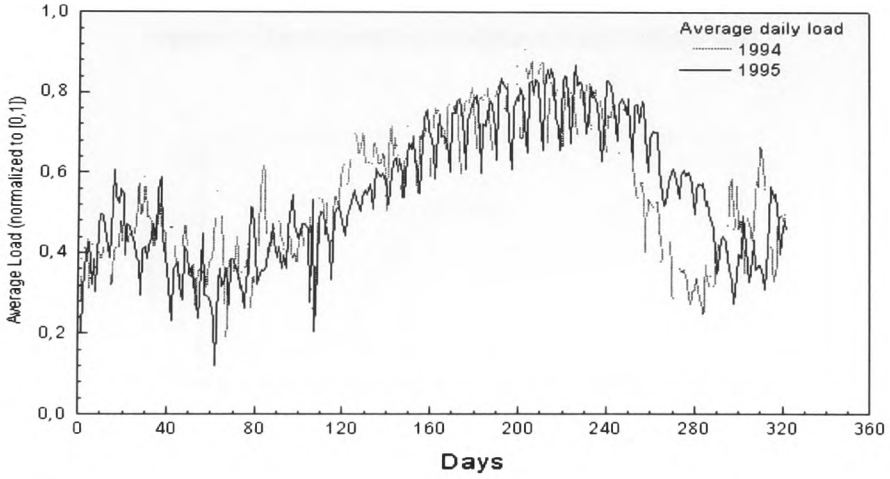


Figure 2: Average Daily Load Demand for the years 1994-1995 Normalised to [0,1]

Data Preprocessing

As in many other applications in load forecasting, it is possible that initial preprocessing might be performed on the data and the results will be the actual inputs to the Neural Network. Furthermore, two types of data can exist a) arithmetic values and b) nominal, *i.e.* discrete values (possible categories of states of an input, for example days of the week, hours, temperature, *etc.*). In order to convert properly nominal values into arithmetic a nonlinear mapping according to their functional relationship is proposed, with the average load demand. For example, the non-linear mapping of the nominal value day of the week to the resulting input is shown in Figure 3, while the non-linear mapping of the nominal value hour to the resulting input is shown respectively in Figure 4.

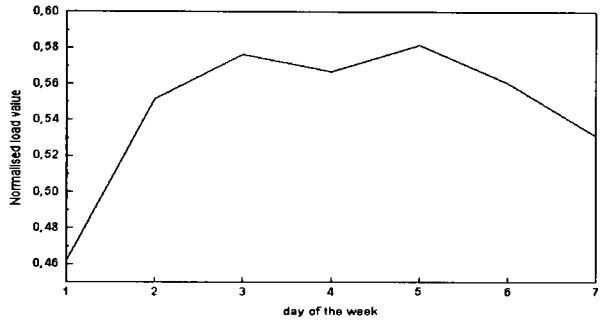


Figure 3: Reconstructed average annual load per day

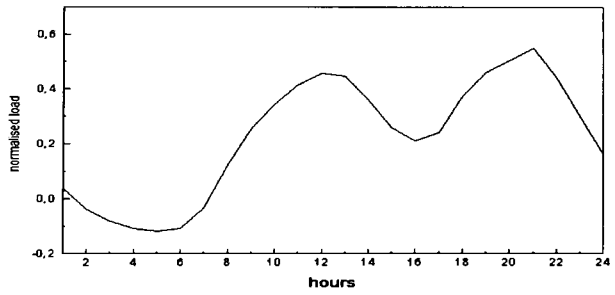


Figure 4: Reconstructed average annual load per hour

In most Neural Network applications for load forecasting [9-16] the output of the Neural Network was a single neuron indicating the exact prediction value. In this paper a different approach is presented, where the output is binary represented in a Gray scale scheme indicating various increments of load prediction.

Networks Description

Based on the existing historical data library, various processing techniques were applied for the construction of the input variables in different experiments. These techniques, basically, examine the embedded periodicity in the load trend. The performance of the system not only depends on the selection of the input patterns but also on the network type, the dimension and the number of the network layers and the learning algorithms used. Various network topologies based on multilayer perceptrons with error back-propagation learning rule were used, in order to identify the architecture that gives the best result. Extensive studies were also performed on the effect of factors such as learning rate, momentum and number of the training patterns.

Two layer perceptrons with the error back-propagation learning rule were used. Various network topologies were examined in order to identify the architecture that gives the best results. Four output neurons were used for the bit-encoded predicted load values. All the experiments were implemented on a Sun Sparc 4 Station with the use of SNNS[19], an academic artificial neural network simulator. The output values ranged from 100MW [0000] to 250MW [1111] and each Gray scale increment covers a range of 10MW. Two year-input patterns were used as training data. The performance of the neural networks depended mainly on the selection of the input variables.

To evaluate the resulting neural network's performance, the following percentage error measure is utilized:

$$\%Error = \left| \frac{actual - forecasted}{actual} \right| \bullet 100$$

Initially a common used data representation [9] was implemented. The input vector consisted of 56 variables and was composed as follows:

- 24 values for the hourly load demand two days before the prediction day.
- 24 values for the hourly load demand one day before the prediction day.
- 4 values for the minimum and maximum temperatures the last two days before the prediction day.
- 2 values for the minimum and maximum predicted temperature values for the prediction day.

- 2 inputs that represent the prediction day and the prediction hour respectively.

Various learning parameters and hidden-layer sizes were used but had a minimal effect on the results. An average forecast error was found to be 3% using the above architecture composed of 56 input, 30 hidden and 4 output neurons. In a second experiment the same input patterns were used for a 24-ouput-neuron network for a whole day forecast without the Gray scale representation. The resultant forecast error was 4.2%.

Reducing the input nodes to 20 resulted in a decrease of the forecast error. The following input variables were used:

- 4 neurons for the load demand for the last four hours before the prediction hour.
- 4 neurons for the load demand of the previous day before the prediction day as follows: the same hour, previous hour, previous two hours, previous three hours of the forecast hour.
- 4 neurons for the hourly load demand of the previous week before the prediction day as follows: the same hour, previous hour, previous two hours, previous three hours of the forecast hour.
- 6 neurons for the equivalent minimum and maximum temperature value one day and one week before the prediction day along with the forecast temperature values at the prediction day.
- One node for the day of the week
- One node for the hour to be forecasted.

The hidden layer consisted of 20 neurons and 4 neurons composed the output layer. A minimum forecast error of 2.1% was achieved using the above architecture. Another important factor that was considered in the daily load curve was the noon and evening load peaks. The hour they appear can yield an efficient approach of the hourly load demand. Various separate neural networks were used for the daily peak load prediction. The most efficient network architecture consisted of 12 variables, is described below.

- 4 nodes for the midday and evening peak load value one day and one week before the prediction.

- 6 nodes for the equivalent minimum and maximum temperature value one day and one week before the forecast together with the forecast temperature values at the prediction day.
- One node for the day of the week.
- One node for the month of the prediction

Considering these variables along with 20 hidden neurons were selected for the hidden and 4 Gray scale encoding neurons for the output layer. The forecast error achieved with this network architecture was 1.8%. Table 2 summarizes the neural network architectures used along with the resulting average forecast error.

Table 2. Load forecast results

Network Topology	Forecast error
56-30-4	3%
56-30-24	4.2%
20-20-4	2.1%
12-12-4 (peaks)	1.8%

Conclusions

A new artificial neural network approach in terms of input preprocessing and output representation proposed for short term load forecasting. A detailed study has been carried out in the representation of the related factors as inputs to the neural networks. A variety of neural network topologies were also examined. The forecast errors in all the cases were very encouraging.

References

[1] G.E. Box & G.M. Jenkins. "Time Series Analysis-Forecasting and control," Holden-Day, San Francisco, 1976.

[2] S. Vemuri, D Hill, and R. Balasubramanian, "Load forecasting using stochastic models," Paper No. TPI-B, Proceeding of 8th PICA Minneapolis, Minnesota , April 1973.

[3] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," IEEE Transactions on Power Apparatus and Systems, Vol.90, pp. 900-910, April 1971.

[4] P. Gupta and Y. Yamada, "Adaptive short-term forecasting of hourly loads using weather information," IEEE Transactions on Power Apparatus and Systems , Vol.91, pp. 2085-2094, 1972.

[5] C. Asbury, "Weather load model for electric demand energy forecasting," IEEE Transactions on Power Apparatus and Systems, Vol.94, pp. 1111-1116, 1975.

[6] S. Rahman and R. Bhatnager, "An expert system based algorithm for short load forecast," IEEE Transactions on Power Systems, Vol.3, pp.392-399, May 1988.

[7] K. Jobbour, J. Riveros, D. Landbergen, and W. Meyer "ALFA: Automated load forecasting assistant," IEEE Transactions on Power Systems Vol.3, pp. 908-914, August 1988.

[8] Emil Pelikan. "Neural Networks in electric load forecasting," Proceeding EuroNet '93, Prague, September, 1993.

[9] S.J. Kiartzis, A.G. Bakirtzis, V. Petridis. "Short term load forecasting using an artificial neural network," Tech. Chorn-B, Greece, Vol.14, No.4.

[10] Yuan-Yih Hsu, Chien-Chuen Yang, "Design of artificial neural network for short-term load forecasting. Part II: Multilayer feedforward networks for peak and valley load forecasting," IEEE proceedings-C, Vol.138, pp.414-418, September 1991.

[11] Shin-Tzo Chen, David C. Yu, A.R. Morhaddamjo. "Weather sensitive short-term load forecasting using nonfully connected artificial neural network," IEEE transactions on power systems, vol. 7, no. 3, August 1992, pp. 1098-1105.

[12] Kung-Long Ho, Yuan-Yih Hsu, Chien-Chuen Yang. "Short Term load forecasting using a multilayer neural network with an adaptive learning algorithm," IEEE Transactions on Power Systems, vol.7, pp.141-149. February 1992.

[13] Azzam-ul-Asar, James R. McDonald. "A specification of Neural Network applications in the load forecasting problem," IEEE Transactions on Control System Technology. Vol.2, pp.135-141, June 1994.

[14] T. Peng, N. Hubele, G. Karady, "Advancement in the application of Neural Networks for short-term load forecasting," Transactions on Power Systems, vol.7, pp.250-257, February 1992.

[15] K. Lee, Y Cha, J. Park, "Short-term load forecasting using an artificial neural network," Transactions on Power Systems, vol.7,pp.124-132,February 1992.

[16] B.Bitzer, F. Rober,T. Papazoglou. "Load forecasting methods for the power system from Crete – a comparison of different short and long term forecasting methods," Universities Power Engineering Conference, Greenwich September 1995.

[17] Th. Goumas and G. Georgocostas, "Experiences in regional energy planning in the Greek Islands: Crete-Lesvos-Cyclades," Proceeding: Energy issues in E.C. islands, Heraklion, Crete, November 1990.

[18] William H. Press, "Numerical recipes - the art of scientific computing". Cambridge University Press, Cambridge, 1988.

[19] SNNS v4.1: Stuttgart Neural Network Simulator, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart, 1995.

References

[1] B.M. Weedy, B.J. Cory, *Electric Power Systems* John Wiley & Sons, West Sussex England 1998

[2] Edison Electric Institute, *Historic Studies of the Electric Industry*, 1975. Subsequent updating from Power Engineering, pp.54, October 1996

[3] Th. Goumas and G. Georgocostas, "Experiences in regional energy planning in the Greek Islands: Crete-Lesvos-Cyclades," *Proceeding: Energy issues in E.C. islands, Heraklion, Crete*, November 1990.

[4] Public power corporation, Crete, Greece, private communication.

[5] G.E. Box & G.M. Jenkins. *Time Series Analysis-Forecasting and control,* Holden-Day, San Francisco, 1976.

[6] S. Vemuri, D Hill, and R. Balasubramanian, "Load forecasting using stochastic models," Paper No. TPI-B, *Proceeding of 8th PICA,* Minneapolis, Minnesota , April 1973.

[7] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, Vol.90, pp. 900-910, April 1971.

[8] S. Vemuri, W. Huang, and D. Nelson, "On-line Algorithms for forecasting hourly load of an electric utility," *IEEE Transactions on Power Apparatus and Systems*, vol., PAS-100, pp.3775-3784, Aug., 1981

[9] N. Naylor, G. Sell, *Linear operator Theory*, New York, Holt, Rinehart and Winston, 1971

[10] K. Jabbour, J. Riveros, D. Landbergen, W. Meyer, "ALFA: Automated load forecasting assistant," *IEEE Transactions on Power Systems*, vol.3, no.3, pp.908-914, August, 1988

[11] A. Sage, G. Husa, "Algorithms for Sequential Adaptive Estimation of Prior Statistics," Proc. of *IEEE Symposium on Adaptive Processes*, State College, Pa., Nov 1969

[12] R. Mehra, "On the Identification of Variance and Adaptive Kalman Filtering," *Proceeding of JACC*, Boulder, Colorado, pp.494-505, 1969

[13] J. Toyoda, M. Chen, Y. Inoue, "An Application of state estimation to short-term load forecasting, Part 1L Forecasting Modeling,", "Part 2: Implementation," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-89, pp-1678-1688, Oct., 1970

[14] P. Gupta and Y. Yamada, "Adaptive short-term forecasting of hourly loads using weather information," *IEEE Transactions on Power Apparatus and Systems* , Vol.91, pp. 2085-2094, 1972.

[15] C. Asbury, "Weather load model for electric demand energy forecasting," *IEEE Transactions on Power Apparatus and Systems*, Vol.94, pp. 1111-1116, 1975.

[16] S. Rahman and R. Bhatnager, "An expert system based algorithm for short load forecast," *IEEE Transactions on Power Systems*, Vol.3, pp.392-399, May 1988.

[17] G. Irisarri, S. Widergren, P. Yehsakul, "On-Line load forecasting for energy control center application," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-101, no.1, pp.71-78, Jan., 1982

[18] J. Davey, J. Saacks, G. Gunningham, K. Priest, "Practical Application of Weather Sensitive Load Forecasting to System Planning," *IEEE Transactions on Power Apparatus and Systems*, vol.PAS-91, pp.971-977, 1972

[19] R. Thompson, "Weather Sensitive Electric Demand and Energy Analysis on a Large Geographically Diverse Power System - Application to Short Term Hourly Electric Demand Forecasting," *IEEE Transactions on Power Apparatus and Systems* vol.PAS-95, no.1, pp.385-393, Jan. 1976

[20] Q. Lu, W. Grady, M. Crawford, G. Anderson, "An adaptive nonlinear predictor with orthogonal Escalator structure for short-term load forecasting," *IEEE Transactions on Power Apparatus and Systems*, vol.4, no.1, pp.158-164, Feb., 1989

[21] C. Asbury, "Weather load model for electric demand energy forecasting," *IEEE Transactions on Power Apparatus and Systems*, Vol.94, pp. 1111-1116, 1975.

[22] Bunn D.W., Farmer E.D. (editors): *Comparative models for electrical load forecasting.* John Wiley & Sons, Chichester, 1985.

[23] Emil Pelikan. "Neural Networks in electric load forecasting," *Proceeding EuroNet '93*, Prague, September, 1993.

[24] S.J. Kiartzis, A.G. Bakirtzis, V. Petridis. "Short term load forecasting using an artificial neural network," *Tech. Chorn-B*, Greece, Vol.14, No.4, 1994.

[25] Yuan-Yih Hsu, Chien-Chuen Yang, "Design of artificial neural network for short-term load forecasting. Part II: Multilayer feedforward networks for peak and valley load forecasting," *IEEE proceedings-C*, Vol.138, pp.414-418, September 1991.

[26] Shin-Tzo Chen, David C. Yu, A.R. Morhaddamjo. "Weather sensitive short-term load forecasting using nonfully connected artificial neural network," *IEEE transactions on power systems*, vol. 7, no. 3, pp. 1098-1105, August 1992

[27] Kung-Long Ho, Yuan-Yih Hsu, Chien-Chuen Yang. "Short Term load forecasting using a multilayer neural network with an adaptive learning algorithm," *IEEE Transactions on Power Systems*, vol.7, pp.141-149. February 1992.

[28] Azzam-ul-Asar, James R. McDonald. "A specification of Neural Network applications in the load forecasting problem," *IEEE Transactions on Control System Technology*. Vol.2, pp.135-141, June 1994.

[29] T. Peng, N. Hubele, G. Karady, "Advancement in the application of Neural Networks for short-term load forecasting," *Transactions on Power Systems*, vol.7, pp.250-257, February 1992.

[30] K. Lee, Y Cha, J. Park, "Short-term load forecasting using an artificial neural network," *Transactions on Power Systems*, vol.7,pp.124-132,February 1992.

[31] B.Bitzer, F. Rober,T. Papazoglou. "Load forecasting methods for the power system from Crete – a comparison of different short and long term forecasting methods," *Universities Power Engineering Conference*, Greenwich September 1995.

[32] D. Park, M. El-Sharkawi, R. Marks II, L. Atlas, M. Damborg, "Electric Load Forecasting Using An Artificial Neural Network", *IEEE Transactions on Power Systems*, vol.6, no.2, May 1991

[33] B.S. Kermanshahi, C.H. Poskar, G. Swift, P. McLaren, W. Pedrycz, W.Buhr, A.Silk, "Artificial Neural Network for Forecasting Daily Loads of A Canadian Electric Utility", Neural Network Applications, *IEEE technology update series*, selected conference papers, Patrick Simpson (editor), Technical Activities Board, 1996.

[34] Thomas Baumann, Alain J. Germond, "Application of the Kohonen Network to Short-Term Load Forecasting", Neural Network Applications, *IEEE technology update series*, selected conference papers, Patrick Simpson (editor), Technical Activities Board, 1996.

[35] G. Lambert Torres, B. Valiqutte, D. Mukhedkar, "Short-term feeder load forecasting: An expert system using fuzzy logic", *IFAC Power systems and power plant control*, Seoul, Korea, 1989.

[36] Young-Il Park and Jong-Keun Park, "An expert system for short term load forecasting by fuzzy decision", *IFAC Power systems and power plant control*, Seoul, Korea, 1989.

[37] Steven K. Rogers, Matthew Kabrisky, *An introduction to biological and artificial neural networks for pattern recognition*, Spie Optical Engineering Press, Washington, 1991

[38] A. Cichocki, R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons, Stuttgart, 1993

[39] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA*, Vol.79, pp. 2554-2558, April 1982.

[40] T. Kohonen, *Self-Organization and Associative Memory* (Second Edition), Springer-Verlag, Berlin, Germany, 1988.

[41] D.E. Rumelhart, J.L. McClelland, *Parallel Distributed Processing: Foundations*, MIT Press, Cambridge, MA, Vol. 1, 1986

[42] Joseph P. Bigus, *Data Mining with Neural Networks*. McGraw-Hill, 1996

[43] Waserman, *Neural computing: theory and practice*, Van Nostrand Reinhold, New York, 1989

[44] Olle Elgerd, *Electric Energy Systems theory,* McGraw Hill, 1971

[45] Widrow B., M.E. Hoff, *Adaptive switching circuits* IRE WESCON Convention Record, pp. 96-104, 1960.

[46] Rosenblatt F., *The Perceptron: A Probabilistic model for information storage and organization in the brain* Physiological Review 65, pp. 386-408,1958

[47] SNNS v4.1: Stuttgart Neural Network Simulator, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart, 1995.

[48] Broomhead D., Lowe, *Multivariate Function Interpolation and Adaptive Networks,* Complex Systems, Vol 2, pp.179-189, 1988.

[49] Wasserman, P.D., *Advanced Methods in Neural Computing,* Von Nostrand Reinhold, New York, 1993.

[50] Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K.. Phoneme Racognition Using Time Delay Neural Networks. *IEEE Transactions on Acoustics, Speech and Singal Processing,* 37:328-339, 1989.